BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images,
Please do not report the images to the
Image Problem Mailbox.

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/30 — A1

(11) International Publication Number: WO 99/23578

(43) International Publication Date: 14 May 1999 (14.05.99)

(21) International Application Number: PCT/GB98/03196

(22) International Filing Date: 27 October 1998 (27.10.98)

(30) Prlority Data:
08/960,540 31 October 1997 (31.10.97) US

(71) Applicant: INTERNATIONAL BUSINESS MACHINES CORPORATION [US/US]; New Orchard Road, Armonk, NY 10504 (US).

(71) Applicant *(for MC only)*: IBM UNITED KINGDOM LIMITED [GB/GB]; P.O. Box 41, North Harbour, Portsmouth, Hampshire PO6 3AU (GB).

(72) Inventors: CASTELLI, Vittorio; Apartment 2-17, 55 North Broadway, White Plains, NY 10601 (US). LI, Chung-Sheng; Apartment 2C, 50 Croton Avenue, Ossining, NY 10562 (US). THAMASIAN, Alexander; 17 Meadowbrook Road, Pleasantville, NY 10570 (US).

(74) Agent: BOYCE, Conor; IBM United Kingdom Limited, Intellectual Property Law, Hursley Park, Winchester, Hampshire SO21 2JN (GB).

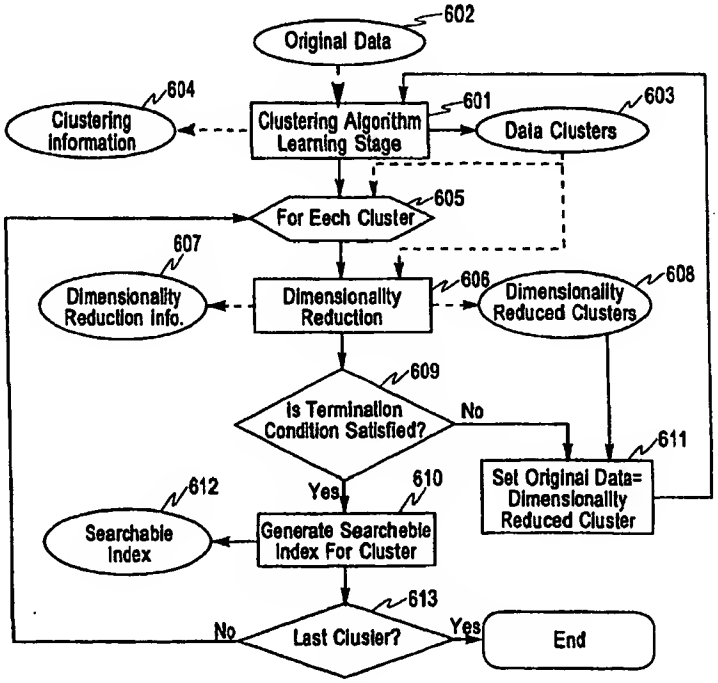
(81) Designated States: CZ, HU, IL, KR, PL, RU, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: MULTIDIMENSIONAL DATA CLUSTERING AND DIMENSION REDUCTION FOR INDEXING AND SEARCHING

(57) Abstract

An improved multidimensional data indexing technique that generates compact indexes such that most or all of the index can reside in main memory at any time. During the clustering and dimensionality reduction, clustering information and dimensionality reduction information are generated for use in a subsequent search phase. The indexing technique can be effective even in the presence of variables which are not highly correlated. Other features provide for efficiently performing exact and nearest neighbor searches using the clustering information and dimensionality reduction information. One example of the dimensionality reduction uses a singular value decomposition technique. The method can also be recursively applied to each of the reduced-dimensionality clusters. The dimensionality reduction can also be applied to the entire database as a first step of the index generation.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LI	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

MULTIDIMENSIONAL DATA CLUSTERING AND DIMENSION REDUCTION FOR INDEXING AND SEARCHING

Field of the Invention

The present invention is related to an improved information system. A particular aspect of the present invention is related to generating and searching compact representations of multidimensional data. A more particular aspect of the present invention is related to generating and searching compact index representations of multidimensional data in database systems using associated clustering and dimension reduction information.

Background

Multidimensional indexing is fundamental to spatial databases, which are widely applicable to Geographic Information Systems (GIS), Online Analytical Processing (OLAP) for decision support using a large data warehouse, and multimedia databases where high-dimensional feature vectors are extracted from image and video data.

Decision support is rapidly becoming a key technology for business success. Decision support allows a business to deduce useful information, usually referred to as a *data warehouse*, from an operational database. While the operational database maintains state information, the data warehouse typically maintains historical information. Users of data warehouses are generally more interested in identifying trends rather than looking at individual records in isolation. Decision support queries are thus more computationally intensive and make heavy use of aggregation. This can result in long completion delays and unacceptable productivity constraints.

Some known techniques used to reduce delays are to pre-compute frequently asked queries, or to use sampling techniques, or both. In particular, applying online analytical processing (OLAP) techniques such as *data cubes* on very large relational databases or data warehouses for decision support has received increasing attention recently (see e.g., Jim Gray, Adam Bosworth, Andrew Layman, and Hamid Pirahesh, "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals", International Conference on Data Engineering, 1996, New Orleans, pp. 152-160) (Gray). Here, users typically view the historical data from data warehouses as multidimensional data cubes. Each cell (or lattice point) in the cube is a view consisting of an aggregation of interests, such as total sales.

Multidimensional indexes can be used to answer different types of queries, including:

- find record(s) with specified values of the indexed columns (exact search);
- find record(s) that are within [a1..a2], [b1..b2], ... , [z1..z2] where *a*, *b* and *z* represent different dimensions (range search); and
- find the *k* most similar records to a user-specified template or example (*k*-nearest neighbor search).

Multidimensional indexing is also applicable to image mining. An example of an image mining product is that trademarked by IBM under the name MEDIAMINER, which offers two tools: Query by Image Content (QBIC); and IMAGEMINER, for retrieving images by analyzing their content, rather than by searching in a manually created list of associated keywords.

QBIC suits applications where keywords cannot provide an adequate result, such as in libraries for museums and art galleries; or in online stock photos for Electronic Commerce where visual catalogs let you search on topics, such as wallpapers and fashion, using colors and texture. Image mining applications such as IMAGEMINER let you query a database of images using conceptual queries like "forest scene", "ice", or "cylinder". Image content, such as color, texture, and contour are combined as simple objects that are automatically recognized by the system.

These simple objects are represented in a knowledge base. This analysis results in a textual description that is then indexed for later retrieval.

During the execution of a database query, the database search program accesses part of the stored data and part of the indexing structure; the amount of data accessed depends on the type of query and on the data provided by the user, as well as on the efficiency of the indexing algorithm. Large databases are such that the data and at least part of the indexing structure reside on the larger, slower and cheaper part of the memory hierarchy of the computer system, usually consisting of one or more hard disks. During the search process, part of the data and of the indexing structure are loaded in the faster parts of the memory hierarchy, such as the main memory and the one or more levels of cache memory. The faster parts of the memory hierarchy are generally more expensive and thus comprise a smaller percentage of the storage capacity of the memory hierarchy. A program that uses instructions and data that can be completely loaded into the one or more levels of cache memory is

faster and more efficient than a process that in addition uses instructions and data that reside in the main memory, which in turn is faster than a program that also uses instruction and data that reside on the hard disks. Technological limitations are such that the cost of cache and main memory makes it too expensive to build computer systems with enough main memory or cache to completely contain large databases.

Thus, there is a need for an improved indexing technique that generates indexes of such size that most or all of the index can reside in main memory at any time; and that limits the amount of data to be transferred from the disk to the main memory during the search process. The present invention addresses such a need.

Several well known spatial indexing techniques, such as R-trees can be used for range and nearest neighbor queries. Descriptions of R-trees can be found, for example, in "R-trees: A Dynamic index structure for spatial searching" by A. Guttman, ACM SIGMOD Conf. on Management of Data, Boston, MA, June, 1994. The efficiency of these techniques, however, deteriorates rapidly as the number of dimensions of the feature space grows, since the search space becomes increasingly sparse. For instance, it is known that methods such as R-Trees are not useful when the number of dimensions is larger than 8, where the usefulness criterion is the time to complete a request compared to the time required by a brute force strategy that completes the request by sequentially scanning every record in the database. The inefficiency of conventional indexing techniques in high dimensional spaces is a consequence of a well-known phenomenon called the "curse of dimensionality" which is described, for instance, in "From Statistics to Neural Networks" NATO ASI Series, vol. 136, Springer-Verlag, 1994, by V. Cherkassky, J. H. Friedman, and H. Wechsles. The relevant consequence of the curse of dimensionality is that clustering the index space into hypercubes is an inefficient method for feature spaces with a higher number of dimensions.

Because of the inefficiency associated with using existing spatial indexing techniques for indexing a high-dimensional feature space, techniques well known in the art exist to reduce the number of dimensions of a feature space. For example, the dimensionality can be reduced either by variable subset selection (also called feature selection) or by singular value decomposition followed by variable subset selection, as taught, for instance by C. T. Chen, Linear System Theory and Design , Holt, Rinehart and Winston, Appendix E, 1984. Variable subset selection is a well known and active field of study in statistics, and numerous methodologies have been proposed (see e.g., Shibata et al. An Optimal Selection of Regression Variables, Biometrika vol. 68, No. 1, 1981, pp. 45-54. These methods are effective in an index generation system only if

many of the variables (columns in the database) are highly correlated. This assumption is in general incorrect in real world databases.

Thus, there is also a need for an improved indexing technique for high-dimensionality data, even in the presence of variables which are not highly correlated. The technique should generate efficient indexes from the viewpoints of memory utilization and search speed. The present invention addresses these needs.

Summary

In accordance with the aforementioned needs, the present invention is directed to a method for generating compact representations of multidimensional data as claimed in claim 1. The present invention has features for generating searchable multidimensional indexes for databases. The present invention has other features for flexibly generating the indexes and for efficiently performing exact and similarity searches. The present invention has still other features for generating compact indexes which advantageously limit the amount of data transferred from disk to main memory during the search process.

One example of an application of the present invention is to multidimensional indexing. Multidimensional indexing is fundamental to spatial databases, which are widely applicable to: Geographic Information Systems (GIS); Online Analytical Processing (OLAP) for decision support using a large data warehouse; and image mining products for mining multimedia databases where high-dimensional feature vectors are extracted from image and video data.

The present embodiment has yet other features for generating and storing a reduced dimensionality index for the reduced dimensionality clusters.

Depending on the exact spatial indexing technique used within each individual cluster, the target vector can then be retrieved by using the corresponding indexing mechanism. For example, conventional multidimensional spatial indexing techniques, including but not limited to the R-tree can be used for indexing within each cluster. Alternatively, the intra-cluster search mechanism could be a brute force or linear scan if no spatial indexing structure can be utilized.

In a preferred embodiment, the dimension reduction step is a singular value decomposition, and the index is searched for a matching reduced dimensionality cluster, based on decomposed specified data. An example of the dimensionality reduction information is a transformation

matrix (including eigenvalues and eigenvectors) generated by a singular value decomposition and selected eigenvalues of the transformation matrix.

Another example of a method for generating multidimensional indexes having features of the present invention includes the steps of: creating a representation of the database to be indexed as a set of vectors, where each vector corresponds to a row in the database and the elements of each vector correspond to the values, for the particular row, contained in the columns for which an index must be generated; the set of vectors is then partitioned using a clustering technique into one or more groups (also called clusters) and clustering information associated with the clustering is also generated and stored; a dimensionality reduction technique is then applied separately to each of the groups, to produce a low-dimensional representation of the elements in the cluster as well as with dimensionality reduction information; and for each reduced-dimensionality cluster, an index is generated using a technique that generates efficient indices for the number of dimensions of the cluster.

According to another feature of the present invention, the method can be applied separately to each of the reduced-dimensionality clusters, and therefore the method becomes recursive. The process of recursively applying both dimension reduction and clustering terminates when the dimensionality can no longer be reduced.

According to another feature of the present invention, a dimensionality reduction step could be applied to the entire database as a first step in the index generation (before the database partitioning step). During the partitioning (also called clustering) and dimensionality reduction steps, clustering and dimensionality reduction information is generated for use in the search phase.

According to still another feature of the present invention, the clustering step could be appropriately selected to facilitate the dimensionality reduction step. This can be accomplished, for instance, by means of a clustering method that partitions the space according to the local covariance structure of the data, instead of minimizing a loss derived from a spatially invariant distance function, such as the Euclidean distance.

The present invention also has features for assessing if other clusters can contain elements that are closer to the specific data than the farthest of the k most similar element retrieved. As is known in the art, clustering information can be used to reconstruct the boundaries of the partitions, and these boundaries can be used to determine if a cluster can contain one of the k nearest neighbors. Those skilled in the art will

appreciate that the cluster boundaries are a simple approximation to the structure of the cluster itself, namely, from the mathematical form of the boundary it is not possible to tell whether there are elements of clusters near any given position on the boundary. As an example consider a case where the database contains two spherical clusters of data, and the two clusters are extremely distant from each other. A reasonable boundary for this case would be a hyperplane, perpendicular to the line joining the centroids of the clusters, and equidistant from the centroids. Since the clusters are widely separated, there is no data point near the boundary. In other cases, the boundary could be very close to large number of elements of both clusters. Accordingly, the present invention also has features to determine if a cluster could contain one or more of the k-nearest neighbors of specified data, using a hierarchy of approximations to the actual geometric structure of each cluster.

In a preferred embodiment, the present invention is embodied as software stored on a program storage device readable by a machine tangibly embodying a program of instructions executable by the machine to perform method steps for generating compact representations of multidimensional data; efficiently performing exact and similarity searches; generating searchable multidimensional indexes for databases; and efficiently performing exact and similarity searches using the indexes.

Brief Description of the Drawings

These and other features and advantages of the present invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, wherein:

Fig. 1 shows an example of a block diagram of a networked client/server system;

Fig. 2 shows an example of the distribution of the data points and intuition for dimension reduction after clustering;

Fig. 3 shows an example of projecting three points in a 3-D space into a 2-D space such that the projection preserves the relative distance between any two points of the three points;

Fig. 4 shows an example of projecting three points in a 3-D space into a 2-D space where the rank of the relative distance is affected by the projection;

Fig. 5 shows an example of a computation of the distance between points in the original space and the projected subspace;

Fig. 6 shows an example of a logic flow for generating the multidimensional indexing from the data in the database;

Fig. 7 shows an example of a logic flow for performing dimensionality reduction of the data;

Fig. 8 shows an example of logic flow for an exact search using the index generated without recursive decomposition and clustering;

Fig. 9 shows an example of logic flow for an exact search using the index generated with recursive decomposition and clustering;

Fig. 10 shows an example of logic flow for a *k*-nearest neighbor search using the index generated without recursive decomposition and clustering;

Fig. 11 shows an example of logic flow for a *k*-nearest neighbor search using the index generated with recursive decomposition and clustering;

Fig 12 shows an example of data in a 3 dimensional space, and a comparison of the results of a clustering technique based on Euclidean distance and of a clustering technique that adapts to the local structure of the data;

Fig. 13 shows an example of logic flow of a clustering technique that adapts to the local structure of the data;

Fig. 14 shows an example of a complex hyper surface in a 3-dimensional space and two successive approximations generated using a 3-dimensional quad tree generation algorithm; and

Fig. 15 shows an example of logic flow of the determination of the clusters that can contain elements closer than a fixed distance from a given vector, using successive approximations of the geometry of the clusters.

Detailed Description

Figure 1 depicts an example of a client/server architecture having features of the present invention. As depicted, multiple clients (101) and multiple servers (106) are interconnected by a network (102). The server (106) includes a database management system (DBMS) (104) and direct access storage device (DASD) (105). A query is typically prepared on the client (101) machine and submitted to the server (106) through the network

(102). The query typically includes specified data, such as a user provided example or search template, and interacts with a database management system (DBMS) (104) for retrieving or updating a database stored in the DASD (105). An example of a DBMS is that sold by IBM under the trademark DB2.

According to one aspect of the present invention, queries needing multidimensional, e.g., spatial indexing (including range queries and nearest neighbor queries) will invoke the multidimensional indexing engine (107). The multidimensional indexing engine (107) (described with reference to Figures 8-11) is responsible for retrieving those vectors or records which satisfy the constraints specified by the query based on one or more compact multidimensional indexes (108) and clustering (111) and dimensionality reduction information (112) generated by the index generation logic (110) of the present invention (described with reference to Figures 6 and 7). Most if not all of the compact indexes (108) of the present invention can preferably be stored in the main memory and/or cache of the server (106). Those skilled in the art will appreciate that a database, such as a spatial database, can reside on one or more systems. Those skilled in the art will also appreciate that the multidimensional indexing engine (107) and/or the index generation logic (110) can be combined or incorporated as part of the DBMS (104). The efficient index generation logic (110) and the multidimensional index engine (also called search logic) can be tangibly embodied as software on a computer program product executable on the server (106).

One example of an application is for stored point-of-sale transactions of a supermarket including geographic coordinates (latitude and longitude) of the store location. Here, the server (106) preferably can also support decision support types of applications to discover knowledge or patterns from the stored data. For example, an online analytical processing (OLAP) engine (103) may be used to intercept queries that are OLAP related, to facilitate their processing. According to the present invention, the OLAP engine, possibly in conjunction with the DBMS, uses the multidimensional index engine (107) to search the index (108) for OLAP related queries. Those skilled in the art will appreciate that the index generation logic (110) of the present invention is applicable to multidimensional data cube representations of a data warehouse. An example of a method and system for generating multidimensional data cube representations of a data warehouse is described in co-pending United States patent application S/N 08/843,290, filed April 14, 1997, entitled "System and Method for Generating Multi-Representations of a Data Cube," by Castelli et al., which is hereby incorporated by reference in its entirety.

Multimedia data is another example of data that benefits from spatial indexing. Multimedia data such as audio, video and images can be stored separately from the meta-data used for indexing. One key component of the meta data that can be used for facilitating the indexing and retrieval of the media data are the *feature vectors* extracted from the raw data. For example, a texture, color histogram and shape can be extracted from regions of the image and be used for constructing indices (108) for retrieval.

An example of an image mining application is QBIC, which is the integrated search facility in IBM's DB2 Image Extender. QBIC includes an image query engine (server), and a sample client consisting of an HTML graphical user interface and related common gateway interface (CGI) scripts that together form the basis of a complete application. Both the server and the client are extensible so that a user can develop an application-specific image matching function and add it to QBIC. The image search server allows queries of large image databases based on visual image content. It features:

Querying in visual media. "Show me images like this one", where you define what "like" means in terms of colors, layout, texture, and so on.

Ranking of images according to similarity to the query image.

Automatic indexing of images, where numerical descriptors of color and texture are stored. During a search, these properties are used to find similar images.

The combination of visual queries with text queries or queries on parameters such as a date.

Similarly, indexes can be generated by first creating a representation of the database to be indexed as a set of vectors, where each vector corresponds to a row in the database and the elements of each vector correspond to the values, for the particular row, contained in the columns for which an index must be generated.

Creating a representation of the database as a set of vectors is well known in the art. The representation can be created by, but is not limited to: creating for each row of the database an array of length equal to the dimensionality of the index to be generated; and copying to the elements of the array, the values contained in the columns, of the corresponding row, for which the index must be generated.

Assuming that the *i*th element of a vector **v** is v_i, the vector **v** can then be expressed as

$$v = [v_1 \ldots v_N] \quad (1)$$

where 'N' is the number of dimensions of the vector that is used for indexing.

The client side can usually specify three types of queries, all of which require some form of spatial indexing, as described in the Background:

(1) *Exact queries*: where a vector is specified and the records or multimedia data that match the vector will be retrieved;
(2) *Range queries*: where the lower and upper limit of each dimension of the vector is specified.
(3) *Nearest neighbor queries*: where the most similar vectors are retrieved based on a similarity measure.

The most commonly used similarity measure between two vectors, **v1** and **v2**, is the Euclidean distance measure, *d*, defined as

$$d^2 = \sum (v1_i - v2_i)^2 \quad (2)$$

Note that it is not necessary for all of the dimensions *i* to participate in the computation of either a range or nearest neighbor query. In both cases, a subset of the dimensions can be specified to retrieve the results.

Figure 2 shows an example of the distribution of the vectors in a multidimensional space. As depicted, a total of three dimensions are required to represent the entire space. However, only two dimensions are required to represent each individual cluster, as cluster 201, 202, and 203 are located on the x-y, y-z, and z-x planes, respectively. Thus, it can be concluded that dimension reduction can be achieved through proper clustering of the data. The same dimensional reduction cannot be achieved by singular value decomposition alone, which can only re-orient the feature space so that the axis in the space coincides with the dominant dimensions (three in this example).

Eliminating one or more dimensions of a vector is equivalent to projecting the original points into a subspace. Equation (2) shows that only those dimensions where the individual elements in the vector are different, need to be computed. As a result, the projection of the vector into a subspace does not affect the computation of the distance, providing those elements that are eliminated do not vary in the original space.

Figure 3 shows an example of a distance computation in the original space and a projected subspace where the projection preserves the relative distance between any two of the three points. As shown, in the original 3-dimensional (3-D) space, a point (301) is more distant from another point (302) than it is from a third point (303). Here, the projections of these points (304, 305 and 306 respectively) into the 2-dimensional (2-D) subspace, preserves the relative distance between points.

Figure 4 shows an example of projecting three points in a 3-D space into a 2-D space where the projection affects the rank of the relative distance. As shown, the distance between points (401) and (402) in the 3-D space is larger than that between points (402) and (403). In this case however, the distance between the respective projected points (404) and (405) is shorter than the projected distance between (405) and (406). Thus, the relative distance between two points may not be preserved in the projected subspace.

In the following, a methodology will be derived to *estimate* the maximum error that can result from projecting vectors into a subspace. The process starts by determining the bound of the maximum error. Denoting the centroid of a cluster as Vc, which is defined as

$$V_c = \frac{1}{N}\sum V_i \qquad (3)$$

where N is the total number of vectors in the cluster, which consists of vectors {V1, ... , VN}. After the cluster is projected into a k dimensional subspace, where without loss of generality the last $(n-k)$ dimensions are eliminated, an error is introduced to the distance between any two vectors in the subspace as compared to the original space. The error term is

$$Error^2 = \sum_{i=k+1}^{n}(V_{1,i} - V_{2,i})^2 \qquad (4)$$

The following inequality immediately hold:

$$\begin{aligned} Error^2 &\le \sum_{i-k+1}^{n} (|V_{1,i}|+|V_{2,i}|)^2 \\ &\le \sum_{i-k+1}^{n} (2\max(|V_{1,i}|,|V_{2,i}|))^2 \\ &\le 4 \sum_{i-k+1}^{n} \max(|V_{1,i}|,|V_{2,i}|)^2 \end{aligned} \tag{5}$$

Equation (5) shows that the maximum error incurred by computing the distance in the projected subspace is bounded.

Figure 5 shows an example of an approximation in computing the distances in accordance with the present invention. The distance between a template point T (501) and a generic point V (506) is given by equation 2. This Euclidean distance is invariant with respect to: rotations of the reference coordinate system; translations of the origin of the coordinate system; reflections of the coordinate axes; and the ordering of the coordinates. Let then, without loss of generality, the point V (506) belong to Cluster 1 in Figure 5. Consider then the reference coordinate system defined by the eigenvectors of the covariance matrix of Cluster 1, and let the origin of the reference coordinate system be Centroid 1. The distance between the template T (501) and a generic point V in cluster 1 (505) can be written as

$$d^2 = \sum_{i=1}^{n} (T_i - V_i)^2 \tag{6}$$

where the coordinates T_i and V_i are relative to the reference system. Next, project the elements of Cluster 1 (505) onto Subspace 1, by setting the last n-k+1 coordinates to 0. The distance between the template T (501) and the projection V' (507) of V (506) onto Subspace 1 is now

$$\begin{aligned} d'^2 &= \sum_{i=1}^{k} (T_i - V_i)^2 + \sum_{i=k+1}^{n} (T_i)^2 \\ &= d_1^2 = d_2^2 \end{aligned} \tag{7}$$

The term d_1 is the Euclidean distance between the projection (504) of the template onto Subspace 1, called Projection 1, and the projection V' (507) of V (506) onto Subspace 1; the term d_2 is the distance between the template T (501) and Projection 1 (504), its projection onto Subspace 1; in other words, d_2 is the distance between the template T (501) and Subspace 1. The approximation introduced can now be bound by substituting equation (7) for equation (6) in the calculation of the distance between the template T (501) and the vector V (506). Elementary geometry teaches that the three points here, T (501), V (506) and V' (507), identify an unique 2-dimensional subspace (a plane). To simplify the discussion, let this plane correspond to the plane (520) shown in Figure 5. Then the distance d defined in equation (6) is equal to the length of the segment joining T (501) and V (506), the distance d' defined in equation (7) is the length of the segment joining T (501) and V' (507). A well known theorem of elementary geometry says that the length of a side of a triangle is longer than the absolute value of the difference of the lengths of the other two sides, and shorter than their sum. This implies that the error incurred in substituting the distance d' defined in equation (7) for the distance d defined in equation (6) is less than or equal to the length of the segment joining V (506) and V' (507); thus, the error squared is bounded by

$$error^2 \leq \sum_{i=k+1}^{n} (V_i)^2 \qquad (8)$$

Figure 6 shows an example of a flow chart for generating a hierarchy of reduced dimensionality clusters and low-dimensionality indexes for the clusters at the bottom of the hierarchy. In step 601, a clustering process takes the original data (602) as input; partitions the data into data clusters (603); and generates clustering information (604) on the details of the partition. Each entry in the original data includes a vector attribute as defined in Equation (1). The clustering algorithm can be chosen among, but it is not limited to, any of the clustering or vector quantization algorithms known in the art, as taught, for example, by Leonard Kauffman and Peter J. Rousseeuw in the book "Finding Groups in Data" John Wiley & Sons, 1990; or by Yoseph Linde, Andres Buzo and Robert M. Gray in "An Algorithm for Vector Quantizer Design" published in the *IEEE Transactions on Communications*, Vol. COM-28, No. 1, January 1980, P. 84-95. The clustering information (604) produced by the learning stage of a clustering algorithm varies with the algorithm; such information allows the classification stage of the algorithm to produce the cluster number of any new, previously unseen sample, and to produce a representative vector for each cluster. Preferably, the clustering information includes the

centroids of the clusters, each associated with a unique label (see e.g., Yoseph Linde, Andres Buzo and Robert M. Gray in "An Algorithm for Vector Quantizer Design", supra). In step 605 a sequencing logic begins which controls the flow of operations so that the following operations are applied to each of the clusters individually and sequentially. Those skilled in the art will appreciate that in the case of multiple computation circuits, the sequencing logic (605) can be replaced by a dispatching logic that initiates simultaneous computations on different computation circuits, each operating on a different data cluster. In step 606, the dimensionality reduction logic (606) receives a data cluster (603) and produces dimensionality reduction information (607) and a reduced dimensionality data cluster (608). In step 609, a termination condition test is performed (discussed below). If the termination condition is not satisfied, steps 601 - 609 can be applied recursively by substituting, in step 611, the reduced dimensionality data cluster (608) for the original data (602) and the process returns to step 601. If in step 610 the termination condition is satisfied, a searchable index (612) is generated for the cluster. In step 613, if the number of clusters already analyzed equals the total number of clusters (603) produced by the clustering algorithm in step 601, the computation terminates. Otherwise, the process returns to step 605. The selection of the number of clusters is usually performed by the user, but there are automatic procedures known in the art; see e.g., Brian Everitt, "Cluster Analysis," Halsted Press, 1973, Chapter 4.2.

An example of the termination condition test of step 609 can be based on a concept of data volume, V(X), defined as

$$V(X) = \sum_{i=1}^{m} n_i$$

where X is a set of records, n_i is the size of the ith record and the sum is over all the elements of X. If the records have the same size S before the dimensionality reduction step 606 and n denotes the number of records in a cluster, then the volume of the cluster is Sn. If S' denotes the size of a record after the dimensionality reduction step 606, then the volume of the cluster after dimensionality reduction is S'n. The termination condition can be tested by comparing the volumes Sn and S'n and terminating if Sn = S'n.

In another embodiment, the termination test condition in step 609 is not present and no recursive application of the method is performed.

Figure 7 shows an example of the dimensionality reduction logic of step 606. As depicted, in step 701, a singular value decomposition logic (701) takes the data cluster (702) as input and produces a transformation matrix (703) and the eigenvalues (704) (or *characteristic values)* of the transformation matrix (703). The columns of the transformation matrix are the eigenvectors (or *characteristic vectors*) of the matrix. Algorithms for singular value decomposition are well known in the art; see e.g., R. A. Horn and C. R. Johnson, "Matrix Analysis" Cambridge University Press (1985). Those skilled in the art will appreciate that the singular value decomposition logic can be replaced by any logic performing a same or equivalent operation. In an alternative embodiment, the singular value decomposition logic can be replaced by a principal component analysis logic, as is also well known in the art.

In step 705, a sorting logic takes the eigenvalues (704) as input and produces eigenvalues sorted by decreasing magnitude (706). The sorting logic can be, any of the numerous sorting logic well known in the art. In step 707, the selection logic, selects a subset of the ordered eigenvalues (706) containing the largest eigenvalues (708) according to a selection criterion. The selection criterion can be, but it is not limited to, selecting the smallest group of eigenvalues, the sum of which is larger than a user-specified percentage of the trace of the transformation matrix (703), where the trace is the sum of the elements on the diagonal, as is known in the art. In this example, the transformation matrix (703) and the selected eigenvalues (708) comprise dimensionality reduction information (607). Alternatively, the selection of the eigenvalues can be based on a precision vs. recall tradeoff (described below).

In step 709, the transformation logic takes the data cluster (702) and the transformation matrix (703) as input; and applies a transformation specified by the transformation matrix (703) to the elements of the data cluster (702) and produces a transformed data cluster (710). In step 711, the selected eigenvalues (708) and the transformed data cluster (710) are used to produce the reduced dimensionality data cluster (712). In a preferred embodiment, the dimensionality reduction is accomplished by retaining the smallest number of dimensions such that the set of corresponding eigenvalues account for at least a fixed percentage of the total variance, where for instance the fixed percentage can be taken to be equal to 95%.

Alternatively, the selection of the eigenvalues can be based on a precision vs. recall tradeoff. To understand precision and recall, it is important to keep in mind that the search operation performed by a method of the current invention can be approximate, (as will be described with

reference to Figures 10-11). Let k be the desired number of nearest neighbors to a template in a database of N elements. Here, since the operation is approximate, a user typically requests a number of returned results greater than k. Let n be the number of returned results; of the n results, only c will be correct, in the sense that they are among the k nearest neighbors to the template. The precision is the proportion of the returned results that are correct, and is defined as

$$\text{precision} = c / n,$$

and recall is the proportion of the correct results that are returned by the search, that is,

$$\text{recall} = c / k.$$

Since precision and recall vary with the choice of the template, their expected values are a better measure of the performance of the system. Thus, both precision and recall are meant as expected values (E) taken over the distribution of the templates, as a function of fixed values of n and k:

$$\text{precision} = E(c)/n,$$
$$\text{recall} = E(c)/k.$$

Note that as the number of returned results n increases, the precision decreases while the recall increases. In general, the trends of precision and recall are not monotonic. Since $E(c)$ depends on n, an efficiency vs. recall curve is often plotted as a parametric function of n. In a preferred embodiment, a requester specifies the desired precision of the search and a lower bound on the allowed recall. Then the dimensionality reduction logic performs the dimensionality reduction based on precision and recall as follows: after ordering the eigenvalues in decreasing order, the dimensionality reduction logic (step 606, Figure 6) removes the dimension corresponding to the smallest eigenvalue, and estimates the resulting precision vs. recall function based on a test set of samples selected at random from the original training set or provided by the user. From the precision vs. recall function, the dimensionality reduction logic derives a maximum value of precision n_{max} for which the desired recall is attained. Then the dimensionality reduction logic iterates the same procedure by removing the dimension corresponding to the next smallest eigenvalue, and computes the corresponding precision for which the desired recall is attained. The iterative procedure is terminated when the computed precision is below the threshold value specified by the user, and the dimensionality reduction logic retains only

the dimensions retained at the iteration immediately preceding the one where the termination condition occurs.

In another embodiment of the present invention, the requester specifies only a value of desired recall, and the dimensionality reduction logic estimates the cost of increasing the precision to attain the desired recall. This cost has two components: one that decreases with the number of dimensions, since computing distances and searching for nearest neighbors is more efficient in lower dimensionality spaces; and an increasing component due to the fact that the number of retrieved results must grow as the number of retained dimensions is reduced to insure the desired value of recall. Retrieving a larger number *n* of nearest neighbors is more expensive even when using efficient methods, since the portion of the search space that must be analyzed grows with the number of desired results. Then the dimensionality reduction logic finds by exhaustive search, the number of dimensions to retain that minimizes the cost of the search for the user-specified value of the recall.

The clustering and singular value decomposition can be applied to the vectors recursively (step 601-611) until a terminating condition (step 609) is reached. One such terminating condition can be that the dimension of each cluster can no longer be reduced as described herein. Optionally, more conventional spatial indexing techniques such as the R-tree can then be applied to each cluster. These techniques are much more efficient for those clusters whose dimension have been minimized. This would thus complete the entire index generation process for a set of high dimensional vectors.

As will be described with reference to Figures 8-15, the present invention also has features for performing efficient searches using compact representations of multidimensional data. Those skilled in the art will appreciate that the search methods of the present invention are not limited to the specific compact representations of multidimensional data described herein.

Figure 8 shows an example of a logic flow for an exact search process based on a searchable index (108 or 612) generated according to the present invention. In this example, the index is generated without recursive application of the clustering and singular value decomposition. An exact search is the process of retrieving a record or records that exactly match a search query, such as a search template. As depicted, in step 802 a query including specified data such as a search template (801) is received by the multidimensional index engine (107) (also called cluster search logic). In step 802, the clustering information (604) produced in step 601 of Figure 6, is used to identify the cluster to which

the search template belongs. In step 803, the dimensionality reduction information (607) produced in step 606 of Figure 6, is used to project the input template onto the subspace associated with the cluster identified in step 802, and produce a projected template (804). In step 805, an intra-cluster search logic uses the searchable index (612) generated in step 610 of Figure 6 to search for the projected template. Note that the simplest search mechanism within each cluster is to conduct a linear scan (or linear search) if no spatial indexing structure can be utilized. On most occasions, spatial indexing structures such as R-trees can offer better efficiency (as compared to linear scan) when the dimension of the cluster is relatively small (smaller than 10 in most cases).

Figure 9 shows another example of a flow chart of an exact search process based on a searchable multidimensional index (108 or 612) generated according to the present invention. Here, the index (108 or 612) was generated with a recursive application of the clustering and dimensionality reduction logic. An exact search is the process of retrieving the record or records that exactly match the search template. As depicted, a query including specified data such as a search template (901) is used as input to the cluster search logic, in step 902, which is analogous to the cluster search logic in step 802 of Figure 8. In step 902, clustering information (604) produced in step (601) of Figure 6 is used to identify the cluster to which the search template (901) belongs. In step 903, (analogous to step 803 of Figure 8) the dimensionality reduction information (607), produced in step 606 of Figure 6, is used to project the input template onto the subspace associated with the cluster identified in step 902, and produce a projected template (904). In step 905, it is determined whether the current cluster is terminal, that is, if no further recursive clustering and singular value decomposition steps were applied to this cluster during the multidimensional index construction process. If the cluster is not terminal, in step 907 the search template (901) is replaced by the projected template (904), and the process returns to step 902. In step 906, if the cluster is terminal, the intra-cluster search logic uses the searchable index to search for the projected template. As noted, the simplest intra-cluster search mechanism is to conduct a linear scan (or linear search), if no spatial indexing structure can be utilized. On most occasions, spatial indexing structures such as R-trees can offer better efficiency (as compared to linear scan) when the dimension of the cluster is relatively small (smaller than 10 in most cases).

The present invention has also features for assessing if other clusters can contain elements that are closer to the specific data than the farthest of the k most similar elements retrieved. As is known in the art, clustering information can be used to reconstruct the boundaries of

the partitions, and these boundaries can be used to determine if a cluster can contain one of the *k* nearest neighbors. Those skilled in the art will appreciate that the cluster boundaries are a simple approximation to the structure of the cluster itself. In other words, from the mathematical form of the boundary it is not possible to tell whether there are elements of clusters near any given position on the boundary. As an example, consider a case where the database contains two spherical clusters of data which are relatively distant from each other. A reasonable boundary for this case would be a hyperplane, perpendicular to the line joining the centroids of the clusters, and equidistant from the centroids. Since the clusters are relatively distant however, there is no data point near the boundary. In other cases, the boundary could be very close to large number of elements of both clusters.

As will be described with reference to Figures 14 and 15, the present invention has features computing and storing, in addition to the clustering information, a hierarchy of approximations to the actual geometric structure of each cluster; and using the approximation hierarchy to identify clusters which can contain elements closer than a fixed distance from a given vector.

Figure 10 shows an example of a flow chart of a *k*-nearest neighbor search process based on an index (612) generated according to the present invention. In this example, the index is generated without recursive application of the clustering and singular value decomposition. The *k*-nearest neighbor search returns the *k* closest entries in the database which match the query. The number (*nr*) of desired matches, *k* (1000) is used in step 1002 to initialize the *k*-nearest neighbor set (1009) so that it contains at most *k* elements and so that it is empty before the next step begins. In step 1003, the cluster search logic receives the query, for example a the search template (1001) and determines which cluster the search template belongs to, using the clustering information (604) produced in step 601 of Figure 6. In step 1004, template is then projected onto the subspace associated with the cluster it belongs to, using the dimensionality reduction information (607). The projection step 1004 produces a projected template (1006) and dimensionality reduction information (1005), that includes an orthogonal complement of the projected template (1006) (defined as the vector difference of the search template (1001) and the projected template (1006)), and the Euclidean length of the orthogonal complement. The dimensionality reduction information (1005) and the projected template (1006) can be used by the intra-cluster search logic, in step 1007, which updates the *k*-nearest neighbor set (1009), using the multidimensional index. Examples of the intra-cluster search logic adaptable according to the present invention include any of the nearest neighbor search methods known in the art; see

e.g., "Nearest Neighbor Pattern Classification Techniques" Belur V. Desarathy, (editor),IEEE Computer Society (1991). An example of an intra-cluster search logic (step 1007) having features of the present invention includes the steps of: first, the squared distance between the projected template (1006) and the members of the cluster in the vector space with reduced dimensionality is computed; the result is added to the squared distance between the search template (1001) and the subspace of the cluster; the final result is defined as the "sum" of the squared lengths of the orthogonal complements, (computed in step (1004) which is part of the dimensionality reduction information (1005):

$$\delta^2(template, element) = D^2(projected\_template, element) + \Sigma \| orthogonal\_complement \|^2$$

If the k-nearest neighbor set (1009) is empty at the beginning of step 1007, then the intra-cluster search fills the k-nearest neighbor set with the k elements of the cluster that are closest to the projected template (1006) if the number of elements in the cluster is greater than k, or with all the elements of the cluster otherwise. Each element of the k-nearest neighbor set is associated with a corresponding mismatch index δ^2.

If the k-nearest neighbor set (1009) is not empty at the beginning of step 1007, then the intra-cluster search logic, in step 1007 updates the k-nearest neighbor set when an element is found whose mismatch index δ^2 is smaller than the largest of the indexes currently associated with elements in the k-nearest neighbor set (1009). The k-nearest neighbor set can be updated by removing the element with largest mismatch index δ^2 from the k-nearest neighbor set (1009) and substituting the newly found element for it.

If the k-nearest neighbor set (1009) contains less than k elements, the missing elements are considered as elements at infinite distance. In step 1008, it is determined whether there is another candidate cluster that can contain nearest neighbors. This step takes the clustering information (604) as input, from which it can determine the cluster boundaries. If the boundaries of a cluster to which the search template (1001) does not belong are closer than the farthest element of the k-nearest neighbor set (1009), then the cluster is a candidate. If no candidate exists, the process terminates and the content of the k-nearest neighbor set (1009) is returned as a result. Otherwise, the process returns to step 1004, where the current cluster is now the candidate cluster identified in step 1008.

Figure 11 shows an example of a flow chart of a k-nearest neighbor search process based on a search index (612) generated according to the present invention. Here, the index is generated with a recursive

application of the clustering and singular value decomposition. The k-nearest neighbor search returns the closest k entries in the database, to specified data in the form of a search template. As depicted, in step 1102, the k-nearest neighbor set is initialized to empty and the number of desired matches, k (1100) is used to initialize the k-nearest neighbor set (1111) so that it can contain at most k elements. In step 1103, the cluster search logic takes the search template (1101) as input and associates the search template to a corresponding cluster, using the clustering information (604) produced in step 601 of Figure 6. In step 1104, the template (1101) is projected onto a subspace associated with the cluster identified in step 1103, based on the dimensionality reduction information (607) produced in step 606 of Figure 6. In step 1104, a projected template (1106) and dimensionality reduction information (1105) are produced. Preferably, the dimensionality reduction information (1105) includes the orthogonal complement of the projected template (1106) (defined as the vector difference of the search template (1101) and the projected template (1106)), and the Euclidean length of the orthogonal complement. In step 1107, it is determined if the current cluster is terminal, that is, if no further recursive steps of clustering and singular value decomposition were applied to the current cluster during the construction of the index. If the current cluster is not terminal, in step 1108 the projected template (1106) is substituted for the search template (1101) and the process returns to step 1103. Otherwise, the dimensionality reduction information (1105) and the projected template (1106) are used by the intra-cluster search logic in step 1109, to update the k-nearest neighbor set (1111), based on the searchable index (612). Examples of intra-cluster search logic which are adaptable according to the present invention include any of the nearest neighbor search methods known in the art; see e.g., "Nearest Neighbor Pattern Classification Techniques" IEEE Computer Society, Belur V. Desarathy (editor), 1991. One example of an intra-cluster search logic (step 1007) logic having features of the present invention includes the steps of: first, the squared distance between the projected template (1006) and the members of the cluster in the vector space with reduced dimensionality is computed; the result is added to the squared distance between the search template (1001) and the subspace of the cluster; the result is defined as the "sum" of the squared lengths of the orthogonal complements, (computed in step (1004) which is part of the dimensionality reduction information (1005)):

$$\delta^2(template,element)=D^2(projected\_template,element)+\Sigma\|orthogonal\_complement\|^2$$

If the k-nearest neighbor set (1111) is empty at the beginning of step 1109, then the intra-cluster search logic fills the k-nearest neighbor set (1111) with either: the k elements of the cluster that are closest to the projected template (1106) if the number of elements in the cluster is greater than k; or with all the elements of the cluster if the

number of elements in the cluster is equal or less than k. Each element of the k-nearest neighbor set (1111) is preferably associated with its corresponding mismatch index δ^2.

If the *k*-nearest neighbor set (1111) is not empty at the beginning of step 1109, then the intra-cluster search logic updates the *k*-nearest neighbor set when an element is found whose mismatch index is smaller than the largest of the indexes currently associated with the elements in the *k*-nearest neighbor set (1111). The update can comprise removing the element with the largest mismatch index δ^2 from the *k*-nearest neighbor set (1111) and substituting the newly found element for it.

If the *k*-nearest neighbor set (1111) contains less than *k* elements, the missing elements are considered as elements at infinite distance. In step 1110, it is determined whether the current level of the hierarchy is the top level (before the first clustering step is applied). If the current level is the top level, then the ends and the content of the *k*-nearest neighbor set (1111) is returned as a result. If the current level is not the top level, then in step 1114, a search is made for a candidate cluster at the current level, that is, a cluster that can contain some of the *k'* nearest neighbors. The search is performed using the dimensionality reduction information (1105) and the clustering information (604). In step 1114, the clustering information (604) is used to determine the cluster boundaries. If the boundaries of a cluster to which the search template (1001) does not belong are closer than the farthest element of the *k*-nearest neighbor set (1111), then the cluster is a candidate. If no candidate exists, then in step 1113, the current level is set to the previous level of the hierarchy and the dimensionality reduction information is updated, and the process returns to step 1110. If a candidate exists, then in step 1115, the template is projected onto the candidate cluster, thus updating the projected template (1106); and the dimensionality reduction information is updated. The process then returns to step 1107.

Figures 12 (a)-12 (c) compares the results of clustering techniques based on similarity alone (for instance, based on the minimization of the Euclidean distance between the element of each cluster and the corresponding centroids as taught by Linde, Buzo and Gray in "An Algorithm for Vector Quantizer Design" supra), with clustering using an algorithm that adapts to the local structure of the data. Figure 12 (a) shows a coordinate reference system (1201) and a set of vectors (1202). If a clustering technique based on minimizing the Euclidean distance between the elements of each cluster and the corresponding centroid is used, a possible result is shown in Figure 12 (b): the set of vectors (1202) is partitioned into three clusters, cluster 1 (1205), cluster 2 (1206) and

cluster 3 (1207) by the hyper planes (1203) and (1204). The resulting clusters contain vectors that are similar to each other, but do not capture the structure of the data, and would result in sub-optimal dimensionality reduction. Figure 12 (c) shows the results of clustering using an algorithm that adapts to the local structure of the data. This results in three clusters, cluster 1 (1208), cluster 2 (1209) and cluster 3 (1210), that better capture the local structure of the data and are more amenable to independent dimensionality reduction.

Figure 13 shows an example of a clustering algorithm that adapts to the local structure of the data. In step 1302, the set of vectors (1301) to be clustered and the desired number of clusters are used to select initial values of the centroid (1303). In a preferred embodiment, one element of the set of vectors is selected at random for each of the desired number *NC* of clusters, using any of the known sampling without replacing techniques. In step 1304, a first set of clusters is generated, for example using any of the known methods based on the Euclidean distance. As a result, the samples are divided into *NC* clusters (1305). In step 1306, the centroids (1307) of each of the *NC* clusters are computed, for instance as an average of the vectors in the cluster. In step 1308, the eigenvalues and eigenvectors (1309) of the clusters (1305) can be computed using the singular value decomposition logic (Figure 7, step 701). In step 1310, the centroid information (1307), the eigenvectors and eigenvalues (1309) are used to produce a different distance metric for each cluster. An example of the distance metric for a particular cluster is the weighted Euclidean distance in the rotated space defined by the eigenvectors, with weights equal to the square root of the eigenvalues.

The loop formed by logic steps 1312, 1313 and 1314 generates new clusters. In step 1312, a control logic iterates step 1313 and 1314 over all the vectors in the vector set (1301). In step 1313, the distance between the selected example vector and each of the centroids of the clusters is computed using the distance metrics (1311). In step 1314, the vector is assigned to the nearest cluster, thus updating the clusters (1305). In step 1315, if a termination condition is reached, the process ends; otherwise the process continues at step 1306. In a preferred embodiment, the computation is terminated if no change in the composition of the clusters occurs between two subsequent iterations.

Figure 14 shows an example of a complex surface (1401) in a 3-dimensional space and two successive approximations (1402, 1403) based on a 3-dimensional quad tree, as taught in the art by Samet, H. in "Region Representation Quadtree from Boundary Codes" Comm. ACM 23,3, pp. 163-170 (March 1980). The first approximation (1402) is a minimal bounding box.

The second approximation (1403) is the second step of a quad tree generation, where the bounding box has been divided into 8 hyper rectangles by splitting the bounding box at the midpoint of each dimension, and retaining only those hyper rectangles that intersect the surface.

In a preferred embodiment, the hierarchy of approximations is generated as a k-dimensional quad tree. An example of a method having features of the present invention for generating the hierarchy of approximations includes the steps of: generating the cluster boundaries, which correspond to a zeroth-order approximation to the geometry of the clusters; approximating the convex hull of each of the clusters by means of a minimum bounding box, thus generating a first-order approximation to the geometry of each cluster; partitioning the bounding box into 2^k hyper rectangles, by cutting it at the half point of each dimension; retaining only those hyper rectangles that contain points, thus generating a second-order approximation to the geometry of the cluster; and repeating the last two steps for each of the retained hyper rectangles to generate successively the third-, fourth-,..., n-th approximation to the geometry of the cluster.

Figure 15 shows an example of logic flow to identify clusters that can contain elements closer than a prescribed distance from a given data point, using the hierarchy of successive approximations to the geometry of the clusters. In one embodiment, the geometry of a cluster is a convex hull of the cluster. In another embodiment, the geometry of a cluster is a connected elastic surface that encloses all the points. This logic can be used in searching for candidate clusters, for example in step 1008 of Figure 10. Referring now to Figure 15, in step 1502, the original set of clusters with their hierarchy of geometric approximations (1501) are input to the process. In step 1502, the candidate set (1505) is initialized to the original set (1501). In step 1506, another initialization step is performed by setting the current approximations to the geometry to zero-th order approximations. In a preferred embodiment, the zero-th order geometric approximations of the clusters are given by the decision regions of the clustering algorithm used to generate the clusters. In step 1507, the distances between the current approximations of the geometry of the cluster and the data point (1503) are computed. All clusters more distant than the candidate set (1505) are discarded and a retained set (1508) of clusters is produced. In step 1509, it is determined if there are better approximations in the hierarchy. If no better approximations exist, the computation is terminated by setting the result set (1512) to be equal to the currently retained set (1508). Otherwise, in step 1510, the candidate set (1505) is set to the currently retained set (1508), the current

geometric approximation is set to the immediately better approximation in the hierarchy, and the process return to step 1507.

CLAIMS

1. A computerized method of representing multidimensional data, comprising the steps of:

a) partitioning the multidimensional data into one or more clusters;

b) generating and storing clustering information for said one or more clusters;

c) generating one or more reduced dimensionality clusters and dimensionality reduction information for said one or more clusters; and

d) storing the dimensionality reduction information.

2. The method of claim 1, further comprising the steps of:

generating and storing a reduced dimensionality index for said one or more reduced dimensionality clusters.

3. The method of claim 1 wherein the data is stored in one of a spatial database or a multimedia database which includes a plurality of data records each containing a plurality of fields, further comprising the steps of:

creating a representation of the database to be indexed as a set of vectors, where each vector corresponds to a row in the database and the elements of each vector correspond to the values, for a particular row, contained in the columns for which the searchable index will be generated; and

said partitioning comprising partitioning the vectors into said one or more clusters.

4. The method of claim 2, further comprising the step of storing the index entirely in the main memory of a computer.

5. The method of claim 2, wherein said generating a reduced dimensionality cluster comprises a singular value decomposition further comprising the steps of:

producing a transformation matrix and eigenvalues of the transformation matrix for said each cluster; and

selecting a subset of the eigenvalues including the largest eigenvalues; wherein the dimensionality reduction information includes the transformation matrix and the subset of the eigenvalues.

6. The method of claim 5, for searching for k most similar records to specified data using the reduced dimensionality index, comprising the steps of:

associating specified data to said one or more clusters, based on stored clustering information;

projecting the specified data onto a subspace for an associated cluster based on stored dimensionality reduction information for the associated cluster;

generating dimensionality reduction information including an orthogonal complement for projected specified data, in response to said projecting;

searching, via the index, for the associated cluster having k most similar records to the projected specified data;

determining if any other associated cluster can include any of k most similar records to the projected specified data; and

repeating said searching on said any cluster that can include any of the k most similar records to the specified data.

7. The method of claim 6, wherein the specified data includes a search template, further comprising the steps of:

said projecting step including projecting the template, using the dimensionality reduction information, onto a subspace associated with the cluster to which it belongs;

generating template dimensionality reduction information for a projected template; wherein said searching, via the index is based on the projected template and the template dimensionality reduction information; and

updating a k-nearest neighbor set of the k most similar records to the search template.

8. The method of claim 5, wherein said selecting a subset of the eigenvalues is a function of a precision and a recall of returned results.

9. The method of claim 2, for searching for k records most similar to specified data, the method for searching comprising the steps of:

identifying a cluster to which specified data belongs, based on the clustering information;

reducing the dimensionality of the specified data based on the dimensionality reduction information for an identified cluster;

generating dimensionality reduction information for reduced dimensionality specified data, in response to said reducing;

searching the multidimensional index, using the dimensionality reduction information, for a reduced-dimensionality version of the cluster to which the specified data belongs;

retrieving via the multidimensional index, the k most similar records in the cluster;

identifying other candidate clusters which can contain records closer to the specified data than the farthest of the k most similar records retrieved;

searching a closest other candidate cluster to the specified data, in response to said determining step; and

repeating said identifying and searching steps for all said other candidate clusters.

10. The method of claim 6 or 9, further comprising the step of:

computing the distances (D) between k nearest neighbors in the version of the cluster and the projected specified data as a function of a mismatch index δ^2 wherein,

$\delta^2(template, element) = D^2(projected\_template, element) + \Sigma \| orthogonal\_complement \|^2$.

11. The method of claim 1, wherein the clustering information includes information on a centroid of said one or more clusters, further comprising the step of associating the centroid with a unique label.

12. The method of claim 1, wherein the dimensionality of the data is > 8.

13. The method of claim 1, for performing an exact search, comprising the steps of:

associating specified data to one of the clusters based on stored clustering information;

reducing the dimensionality of the specified data based on stored dimensionality reduction information for a reduced dimensionality version of a cluster, in response to said associating step; and

searching, based on reduced dimensionality specified data, for the reduced dimensionality version of the cluster matching the specified data.

14. The method of claim 13, wherein said searching further comprises the step of conducting a linear scan to match the specified data.

15. The method of claim 1, further comprising the steps of:

creating a hierarchy of reduced dimensionality clusters by recursively applying said steps a) through d); and

generating and storing one or more low-dimensional indexes for clusters at a lowest level of said hierarchy.

16. The method of claim 15, for performing an exact search, comprising the steps of:

recursively applying the steps of:

finding a cluster to which specified data belongs, using stored clustering information; and

reducing the dimensionality of the specified data using stored dimensionality reduction information, until a corresponding lowest level of the hierarchy of reduced dimensionality clusters has been reached; and

searching, using the low dimensional indexes, for a reduced dimensionality version of the cluster matching the specified data.

17. The method of claim 15, for performing a similarity search, further comprising the steps of:

recursively applying the steps of:

finding the cluster to which specified data belongs, using stored clustering information; and

reducing the dimensionality of the specified data to correspond to the lowest level of the hierarchy of reduced dimensionality clusters, using stored dimensionality reduction information;

searching for candidate terminal clusters that can contain one or more of k nearest neighbors of the specified data at each level of the hierarchy of reduced dimensionality clusters starting from a terminal cluster at a lowest level of said hierarchy to which the specified data belongs; and

for each candidate terminal cluster, performing an intra-cluster search for the k nearest neighbors to the specified data.

18. The method of claim 15, for performing a similarity search, further comprising the steps of:

reducing the dimensionality of the specified data;

recursively applying the steps of: finding the cluster to which reduced dimensionality specified data belongs, using stored clustering information; and reducing the dimensionality of the reduced dimensionality specified data to correspond to a lowest level of a hierarchy of reduced dimensionality clusters, using stored dimensionality reduction information;

searching for candidate terminal clusters that can contain one or more of k nearest neighbors of the reduced dimensionality specified data at each level of the hierarchy of reduced dimensionality clusters starting from a terminal cluster at a lowest level of said hierarchy to which the specified data belongs; and

for each candidate terminal cluster, performing an intra-cluster search for the k nearest neighbors to the reduced dimensionality specified data.

19. The method of claim 1, wherein the data is stored in a database, further comprising the steps of:

reducing a dimensionality of the database and generating dimensionality reduction information associated with the database; and

storing the dimensionality reduction information associated with the database;

wherein said partitioning step is responsive to said reducing step.

20. The method of claim 19, for performing an exact search, comprising the steps of:

reducing the dimensionality of specified data, based on the dimensionality reduction information for the database;

associating reduced dimensionality specified data to one of the clusters, based on the clustering information, in response to said reducing;

reducing a dimensionality of the specified data to that of a reduced dimensionality cluster defined by an associated cluster, based on dimensionality reduction information for the associated cluster; and

searching for a matching reduced dimensionality cluster, based on a reduced dimensionality version the specified data.

21. The method of claim 19, for performing a similarity search, comprising the steps of:

reducing the dimensionality of specified data using the dimensionality reduction information associated with the database;

finding the cluster to which reduced dimensionality specified data belongs, based on the clustering information;

reducing the dimensionality of the reduced dimensionality specified data based on the dimensionality reduction information for an identified cluster;

searching for a reduced-dimensionality version of the cluster to which the further reduced dimensionality specified data belongs;

retrieving via the multidimensional index, k records most similar to the further reduced dimensionality specified data in the cluster;

assessing whether other clusters can contain records closer to the specified data than the farthest of the k records retrieved;

searching a closest other cluster to the specified data, in response to said assessing step; and

repeating said assessing and searching for all said other clusters.

22. The method of claim 19, wherein the data is stored in a database, further comprising the step of: generating and storing one or more reduced dimensionality searchable indexes for said one or more reduced dimensionality clusters.

23. The method of claim 19, for performing an exact search, comprising the steps of:

associating specified data to one of the clusters based on stored clustering information;

decomposing the specified data into a reduced dimensionality cluster defined by an associated cluster and stored dimensionality reduction information for the associated cluster, in response to said associating; and

searching said indexes for a matching reduced dimensionality cluster based on decomposed specified data.

24. The method of claim 23, wherein the query includes a search template, further comprising the steps of:

said associating comprising identifying the search template with a cluster based on the stored clustering information;

said decomposing comprising projecting the search template onto a subspace for an identified cluster based on the stored dimensionality reduction information; and

said searching comprising performing an intra-cluster search for a projected template.

25. The method of claim 1, further comprising the steps of:

(a) generating cluster boundaries, which correspond to a zeroth-order approximation to a geometry of said clusters;

(b) approximating the geometry of each of the clusters by means of a minimum bounding box and generating therefrom a first-order approximation to the geometry of each cluster;

(c) partitioning the bounding box into 2k hyper rectangles, wherein said partitioning is at a midpoint of each dimension;

(d) retaining only those hyper rectangles that contain data points and generating therefrom a second-order approximation to the geometry of the cluster; and

(e) repeating said steps (c) and (d) for each retained hyper rectangle to generate successively the third-, fourth-,..., n-th approximations to the geometry of the cluster.

26. The method of claim 25, for searching a hierarchy of approximations to the geometric structure of each cluster, further comprising the steps of:

reducing the dimensionality of the specified data using the dimensionality reduction information associated with the database;

finding the cluster to which reduced dimensionality specified data belongs, based on the clustering information;

reducing the dimensionality of the reduced dimensionality specified data based on the dimensionality reduction information for an identified cluster;

searching for a reduced-dimensionality version of the cluster to which the further reduced dimensionality specified data belongs;

retrieving via the multidimensional index, k records most similar to the further reduced dimensionality specified data in the cluster;

assessing whether one or more other clusters can contain records closer to the specified data than the farthest of the k records retrieved;

retaining an other cluster only if it can contain any of k-nearest neighbors of specified data based on boundaries of the cluster;

iteratively determining if a retained cluster could contain any of the k-nearest neighbors, based on increasingly finer approximations to the geometry, and retaining the retained cluster only if the cluster is accepted at the finest level of the hierarchy of successive approximations; and

identifying a retained cluster as a candidate cluster containing one or more of the k-nearest neighbors of the data, in response to said step of iteratively determining.

27. A program storage device readable by a machine which includes one or more reduced dimensionality indexes to multidimensional data, the program storage device tangibly embodying a program of instructions executable by the machine to perform method steps for representing multidimensional data as claimed in claim 1.

28. A computer program product comprising:

a computer usable medium having computer readable program code means embodied therein for representing multidimensional data, the computer readable program code means in said computer program product comprising:

computer readable program code clustering means for causing a computer to effect partitioning the multidimensional data into one or more clusters;

computer readable program code means, coupled to said clustering means, for causing a computer to effect generating and storing clustering information for said one or more clusters;

computer readable program code dimensionality reduction means, coupled to said clustering means, for causing a computer to effect, generating one or more reduced dimensionality clusters and dimensionality reduction information for said one or more clusters; and

computer readable program code means, coupled to said dimensionality reduction means, for causing a computer to effect storing the dimensionality reduction information.

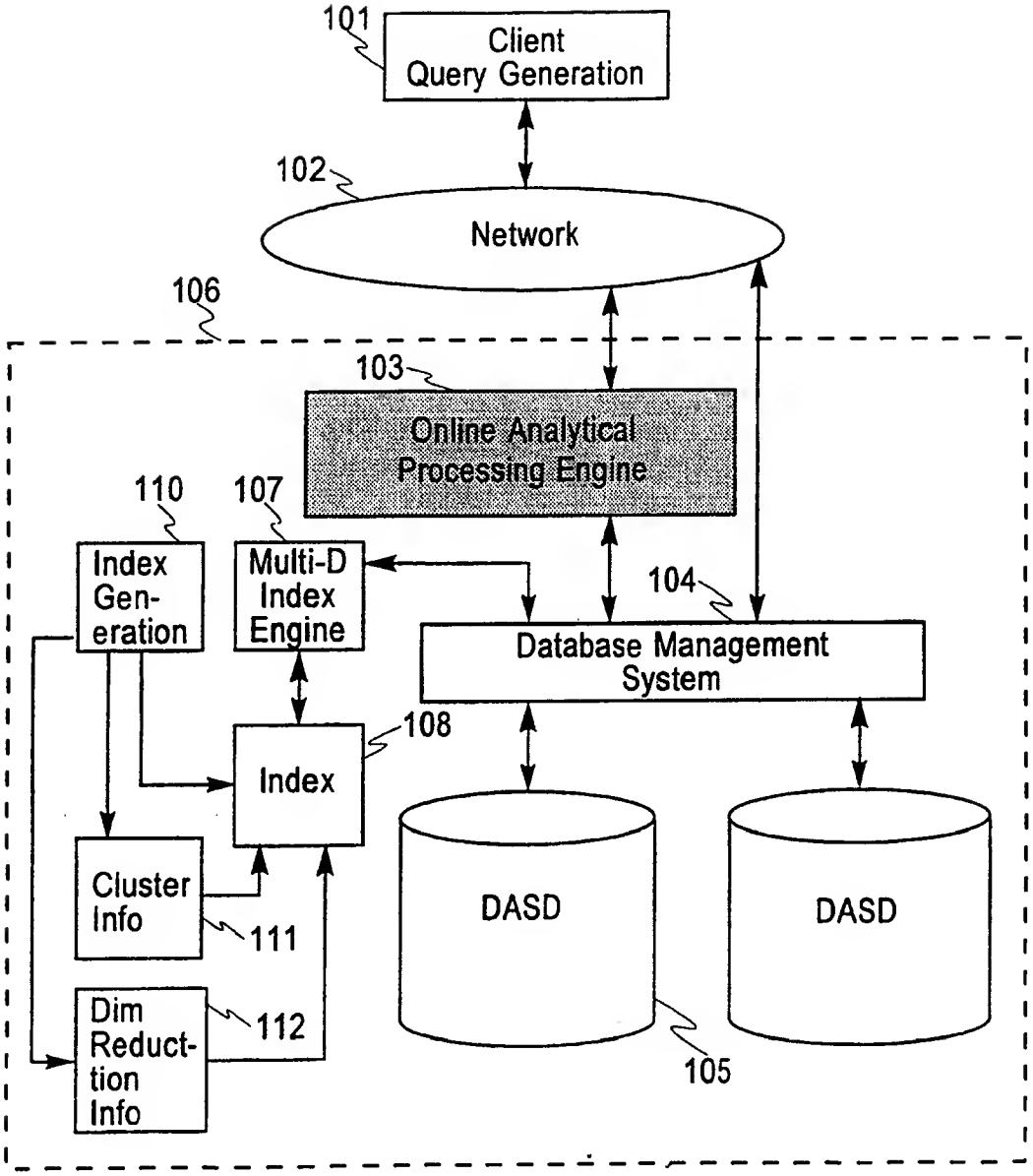

101
Client
Query Generation
102
Network
106
103
Online Analytical
Processing Engine
110
107
Index
Gen-
eration
Multi-D
Index
Engine
104
Database Management
System
108
Index
Cluster
Info
111
Dim
Reduct-
tion
Info
112
DASD
DASD
105

Fig. 1

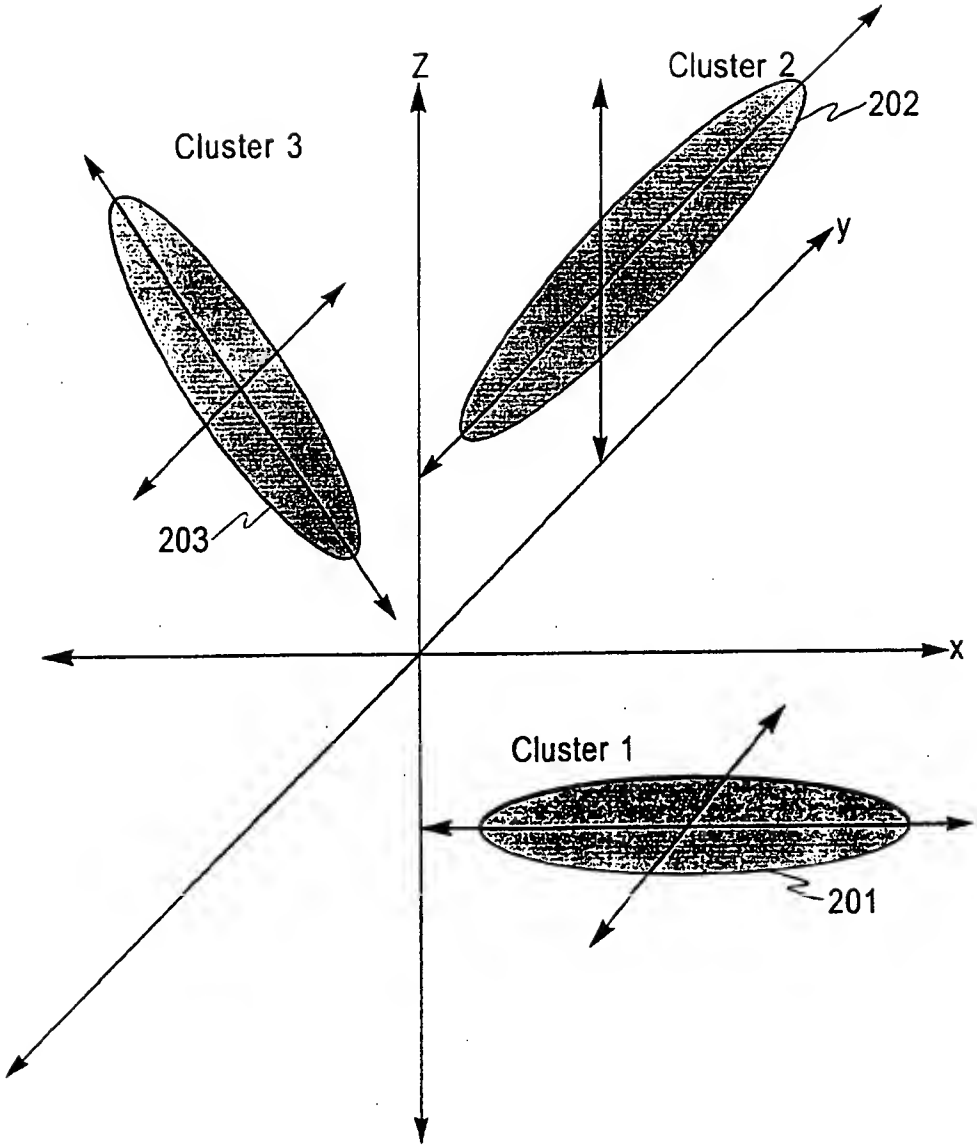


Fig. 2

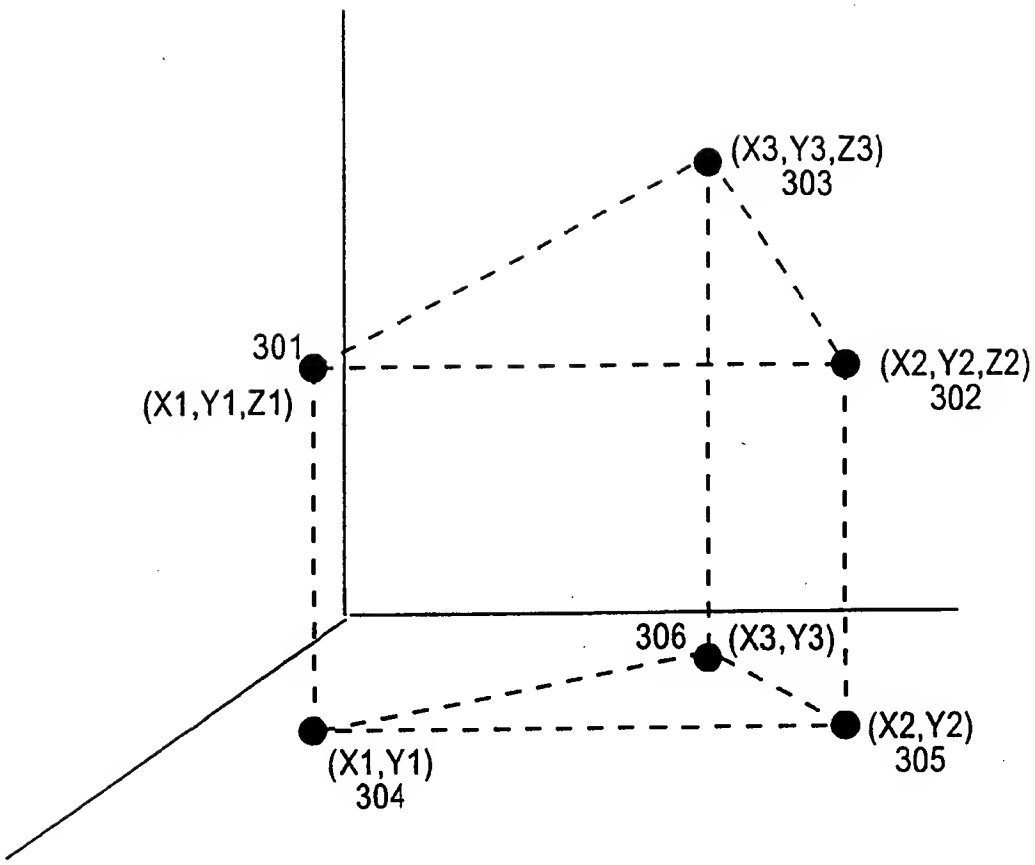


Fig. 3

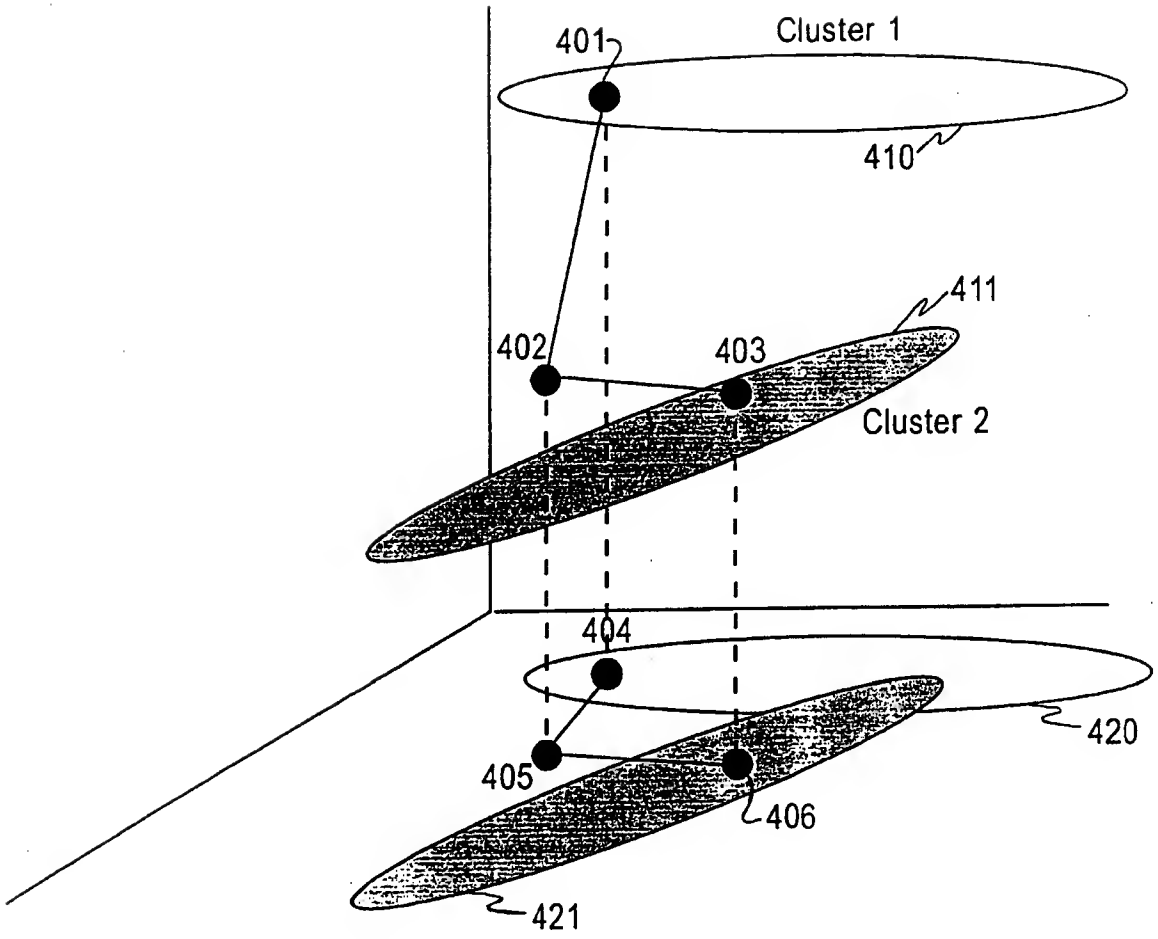


Fig. 4

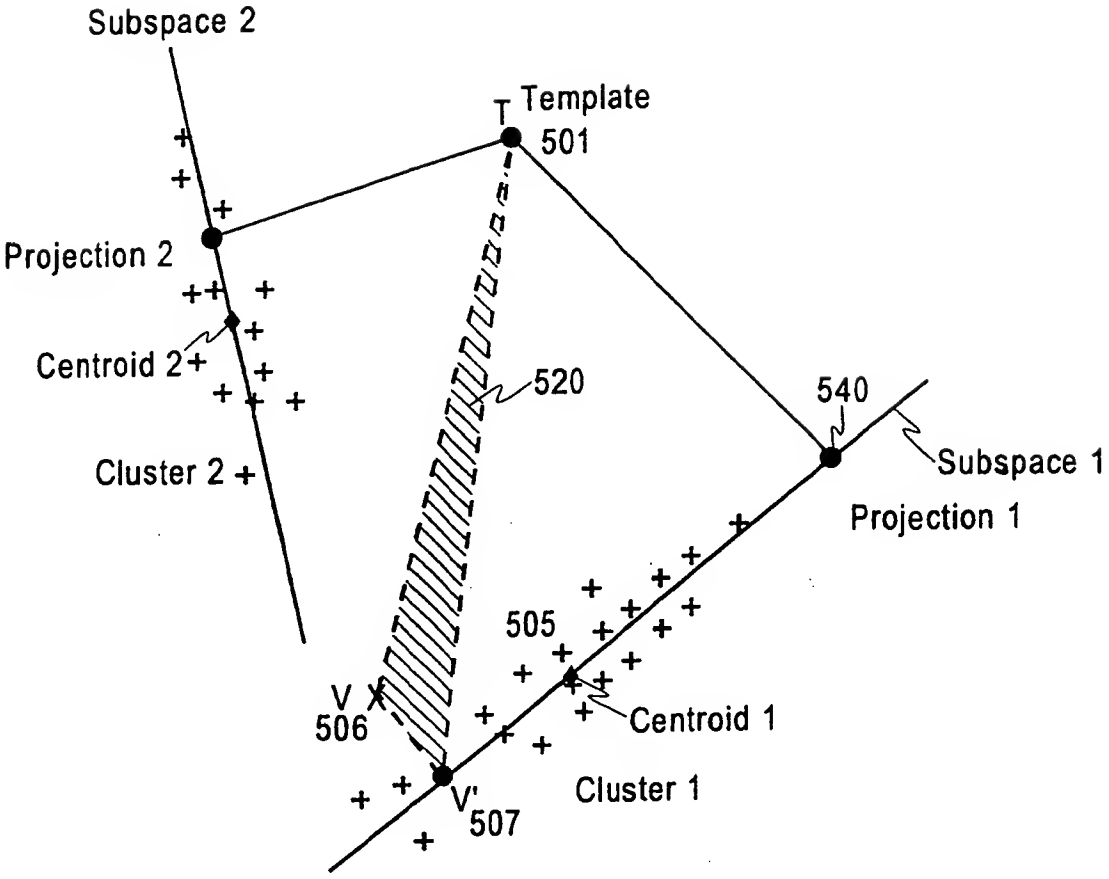

Subspace 2
Template
501
Projection 2
Centroid 2
520
540
Subspace 1
Cluster 2
Projection 1
505
V
506
Centroid 1
V'
507
Cluster 1

Fig. 5

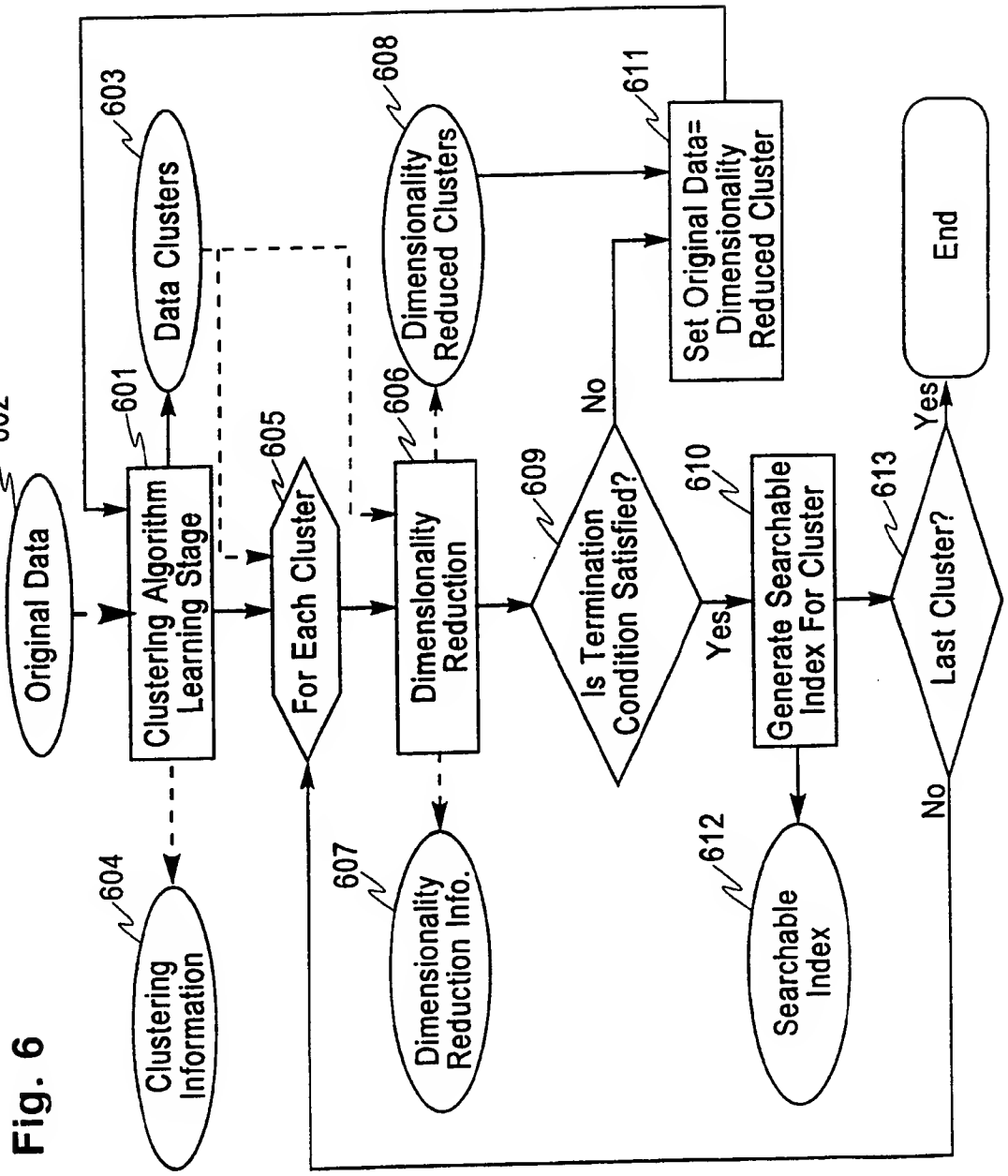

Original Data
602
Clustering Algorithm Learning Stage
601
Data Clusters
603
Clustering Information
604
For Each Cluster
605
Dimensionality Reduction
606
Dimensionality Reduction Info.
607
Dimensionality Reduced Clusters
608
Is Termination Condition Satisfied?
609
No
Yes
Set Original Data= Dimensionality Reduced Cluster
611
Generate Searchable Index For Cluster
610
Searchable Index
612
Last Cluster?
613
Yes
No
End

Fig. 6

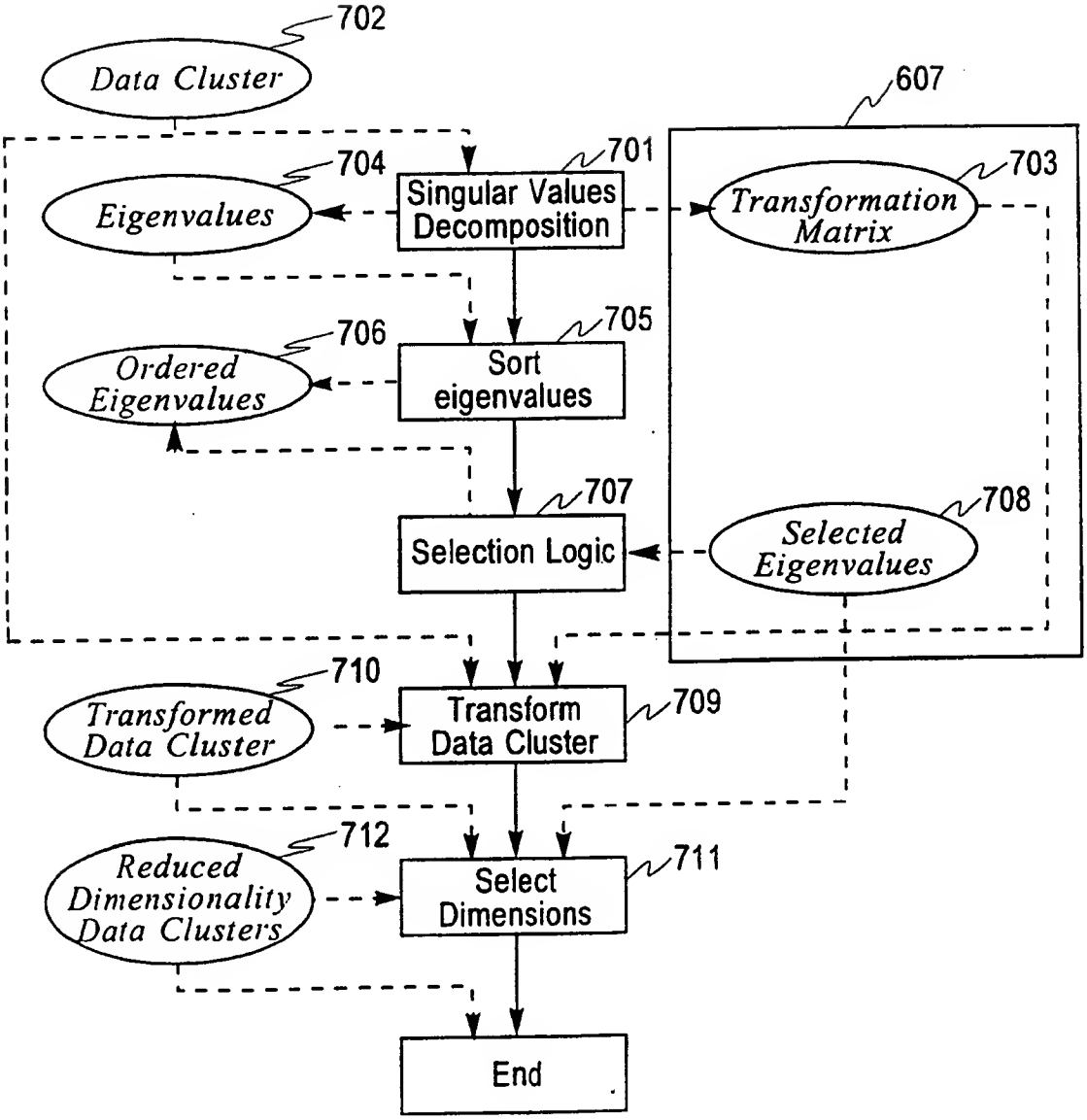

702
Data Cluster
607
701
704
Eigenvalues
Singular Values Decomposition
703
Transformation Matrix
705
706
Ordered Eigenvalues
Sort eigenvalues
707
708
Selection Logic
Selected Eigenvalues
710
Transformed Data Cluster
Transform Data Cluster
709
712
Reduced Dimensionality Data Clusters
Select Dimensions
711
End

Fig. 7

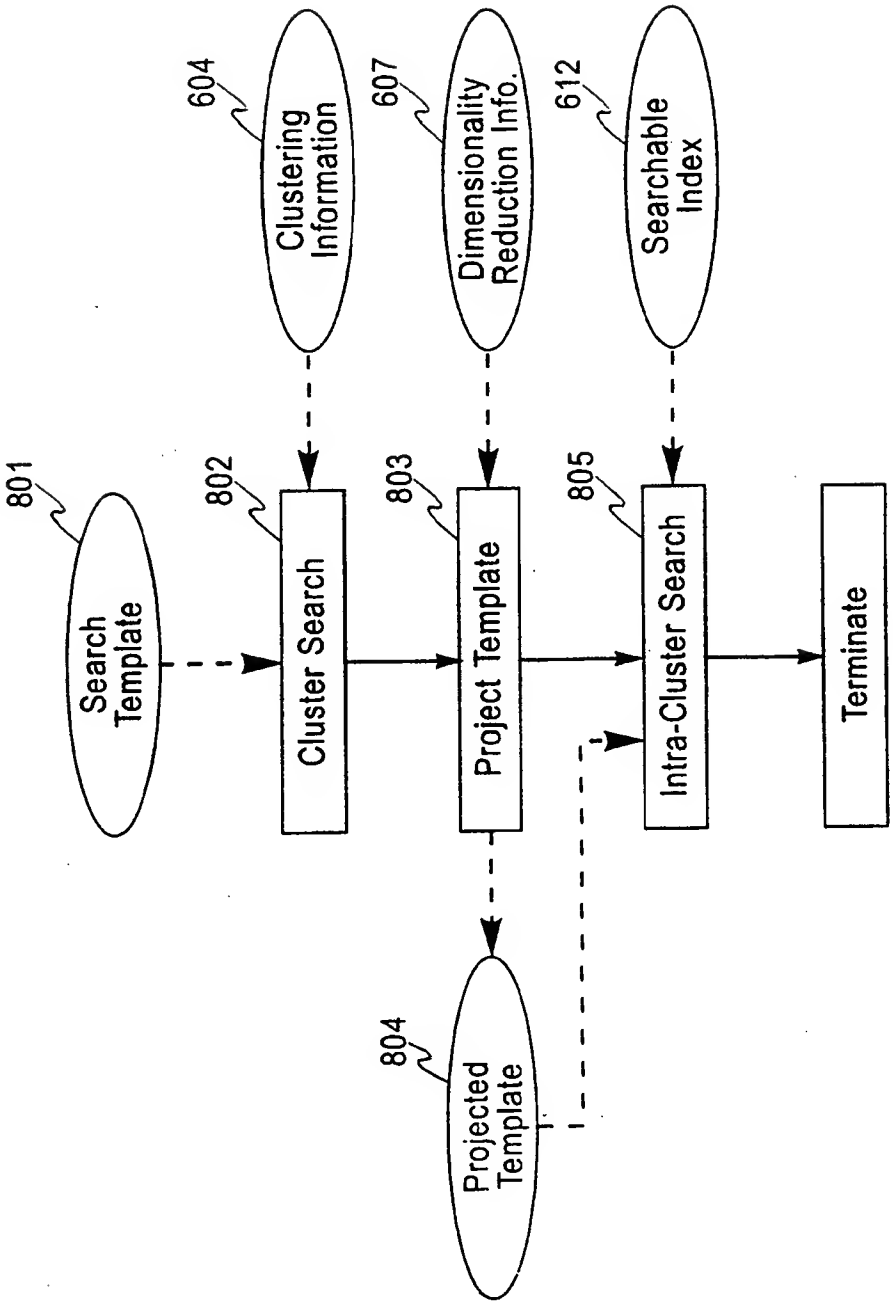

604
Clustering Information
607
Dimensionality Reduction Info.
612
Searchable Index
801
Search Template
802
Cluster Search
803
Project Template
805
Intra-Cluster Search
Terminate
804
Projected Template

Fig. 8

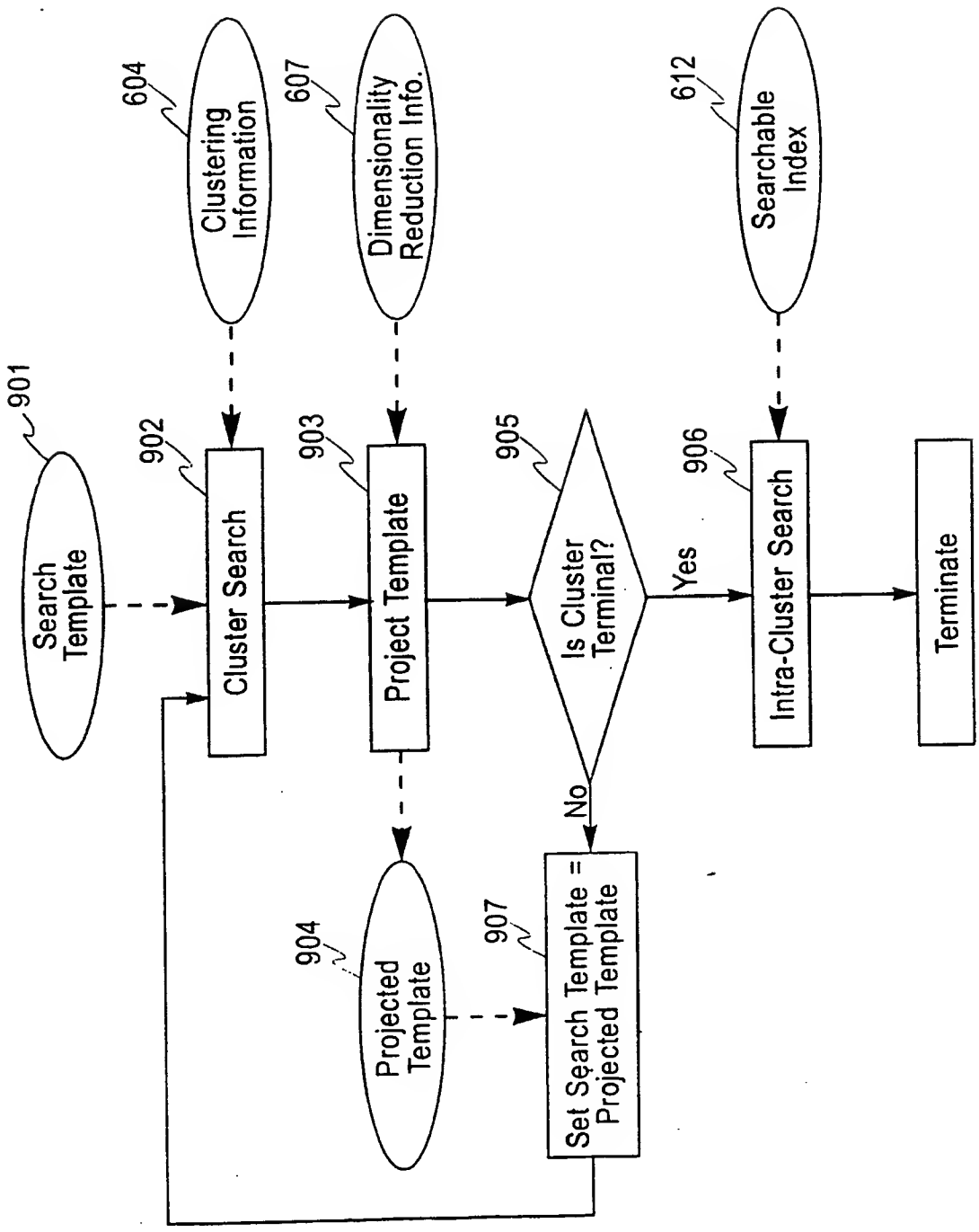

901
Search Template
902
Cluster Search
604
Clustering Information
903
Project Template
607
Dimensionality Reduction Info.
904
Projected Template
905
Is Cluster Terminal?
Yes
No
906
Intra-Cluster Search
612
Searchable Index
907
Set Search Template = Projected Template
Terminate

Fig. 9

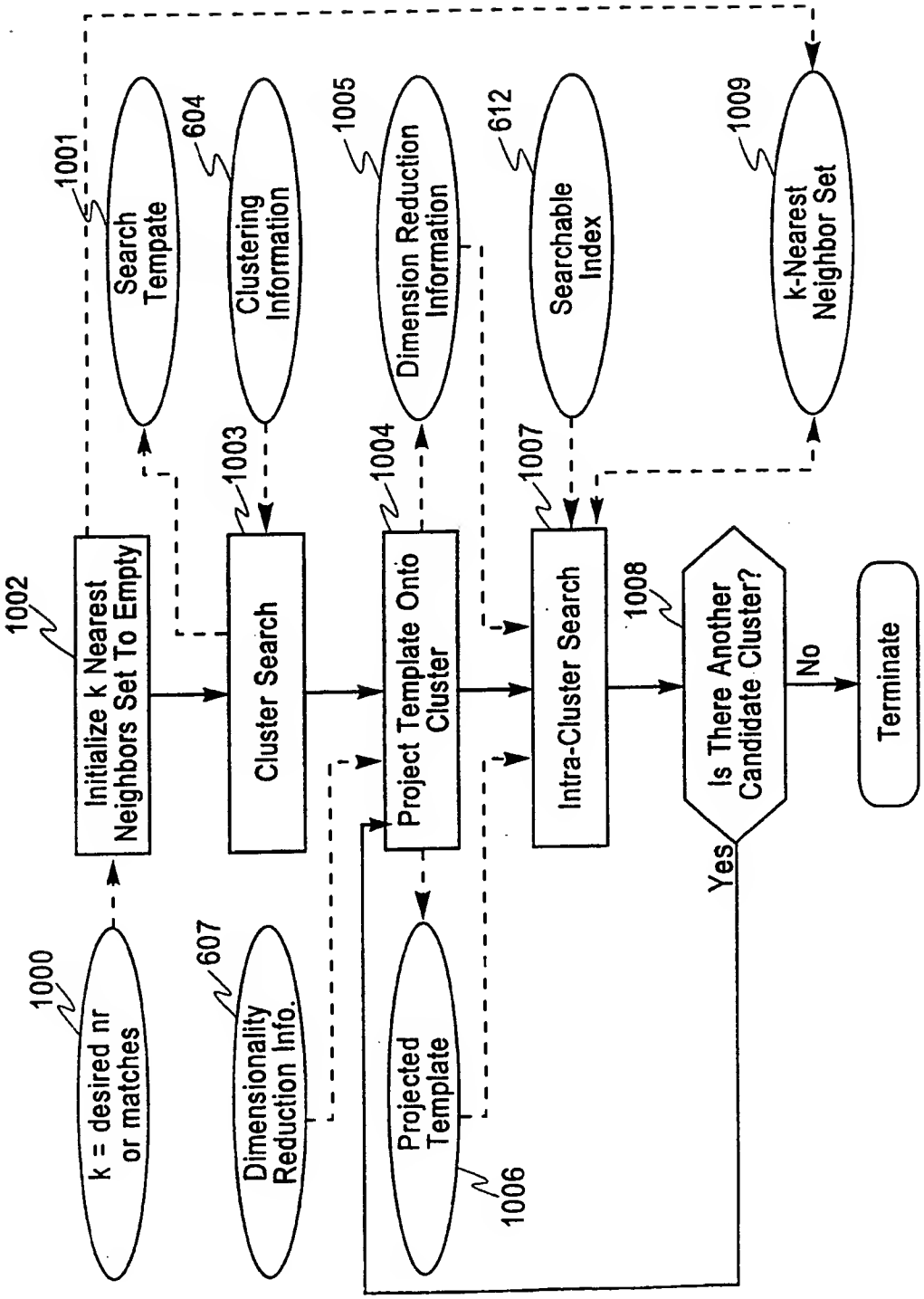

1000
k = desired nr or matches
1001
Search Tempate
1002
Initialize k Nearest Neighbors Set To Empty
1003
Cluster Search
604
Clustering Information
607
Dimensionality Reduction Info.
1004
Project Template Onto Cluster
1005
Dimension Reduction Information
1006
Projected Template
1007
Intra-Cluster Search
612
Searchable Index
1008
Is There Another Candidate Cluster?
Yes
No
1009
k-Nearest Neighbor Set
Terminate

Fig. 10

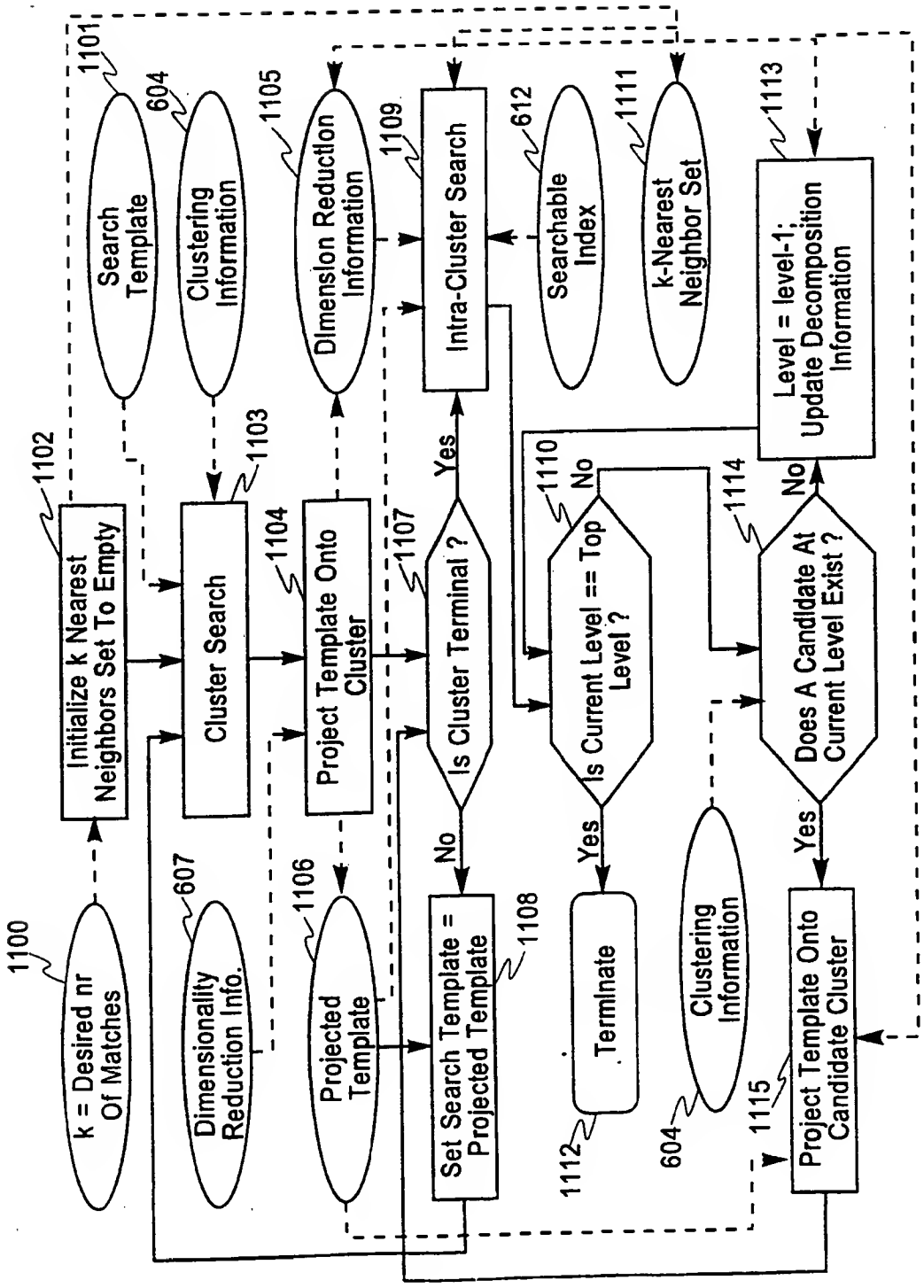

1100
k = Desired nr Of Matches
1101
Search Template
1102
Initialize k Nearest Neighbors Set To Empty
1103
Cluster Search
604
Clustering Information
607
Dimensionality Reduction Info.
1104
Project Template Onto Cluster
1105
Dimension Reduction Information
1106
Projected Template
1107
Is Cluster Terminal ?
Yes
No
1108
Set Search Template = Projected Template
1109
Intra-Cluster Search
612
Searchable Index
1110
Is Current Level == Top Level ?
Yes
No
1112
Terminate
1111
k-Nearest Neighbor Set
1113
Level = level-1; Update Decomposition Information
1114
Does A Candidate At Current Level Exist ?
Yes
No
604
Clustering Information
1115
Project Template Onto Candidate Cluster

Fig. 11

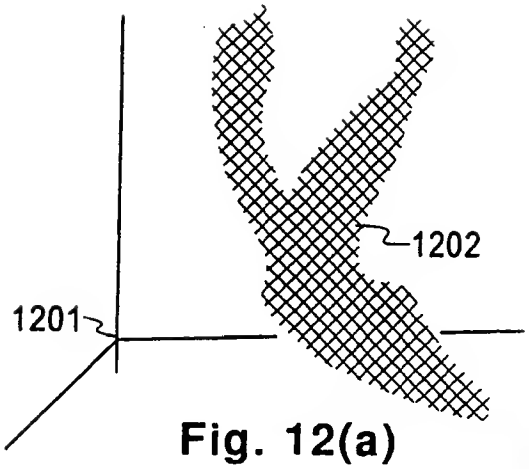


Fig. 12(a)

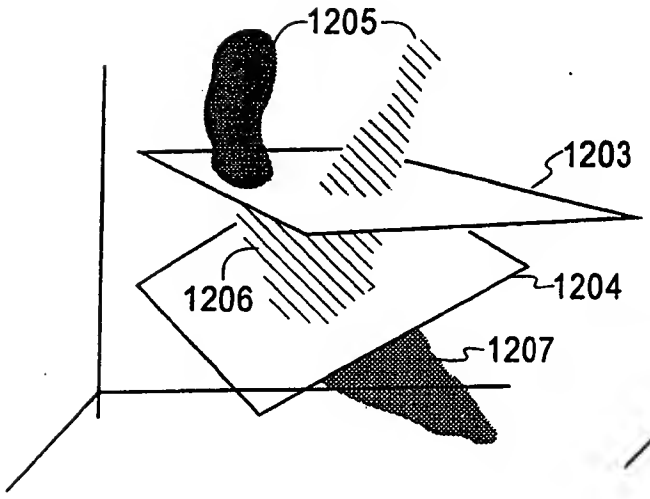


Fig. 12(b)

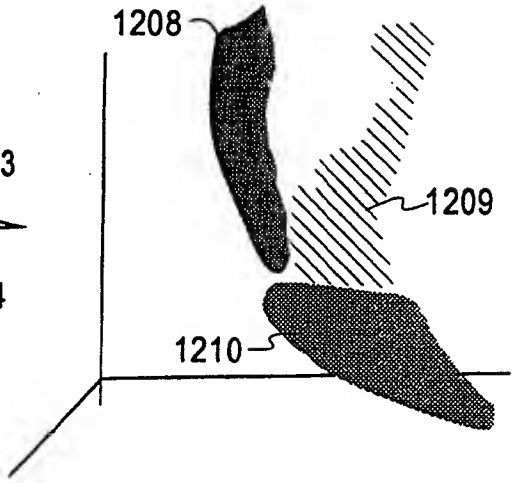


Fig. 12(c)

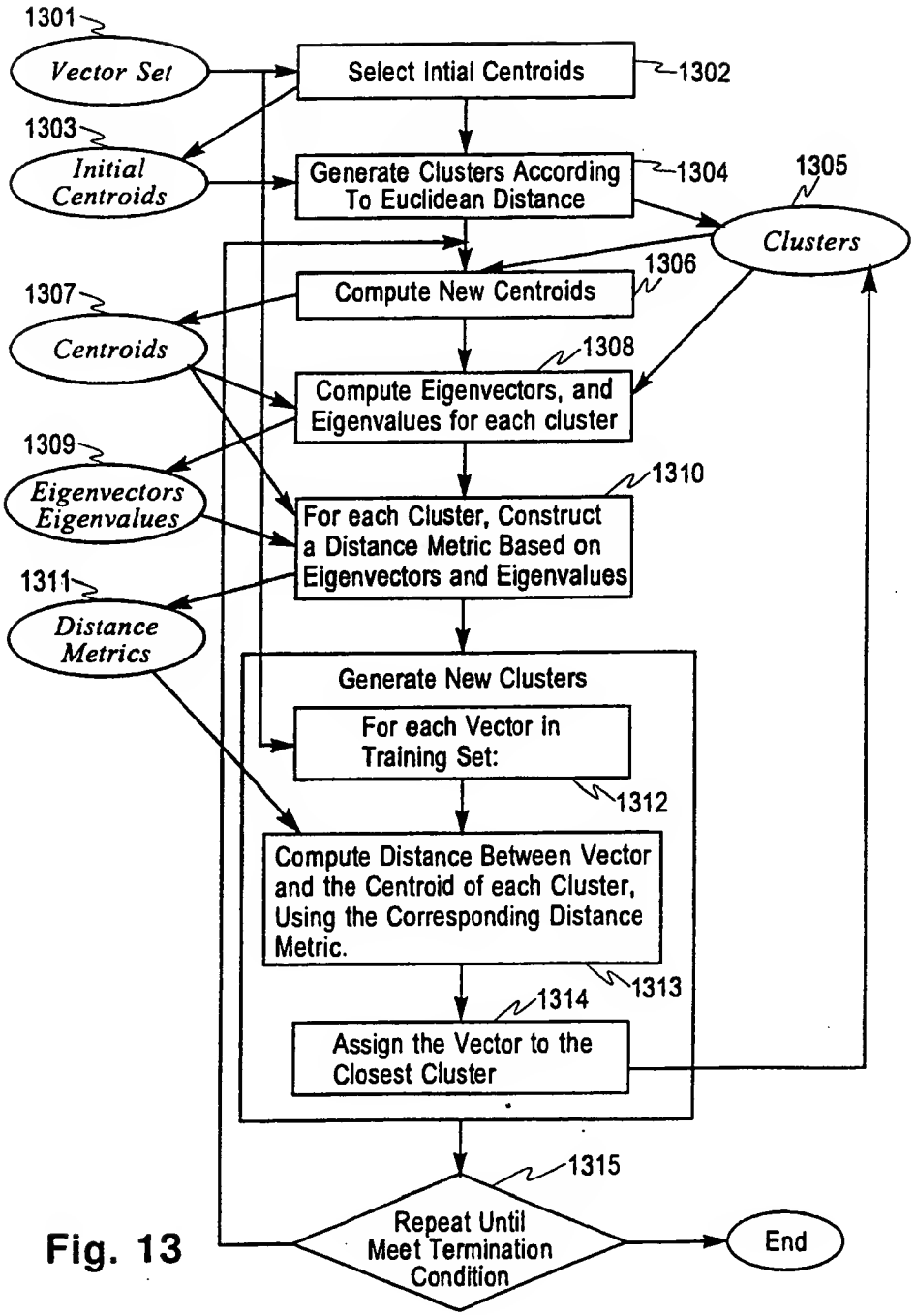

1301
Vector Set
Select Intial Centroids
1302
1303
Initial Centroids
Generate Clusters According To Euclidean Distance
1304
1305
Clusters
1306
Compute New Centroids
1307
Centroids
1308
Compute Eigenvectors, and Eigenvalues for each cluster
1309
Eigenvectors Eigenvalues
1310
For each Cluster, Construct a Distance Metric Based on Eigenvectors and Eigenvalues
1311
Distance Metrics
Generate New Clusters
For each Vector in Training Set:
1312
Compute Distance Between Vector and the Centroid of each Cluster, Using the Corresponding Distance Metric.
1313
1314
Assign the Vector to the Closest Cluster
1315
Repeat Until Meet Termination Condition
End

Fig. 13

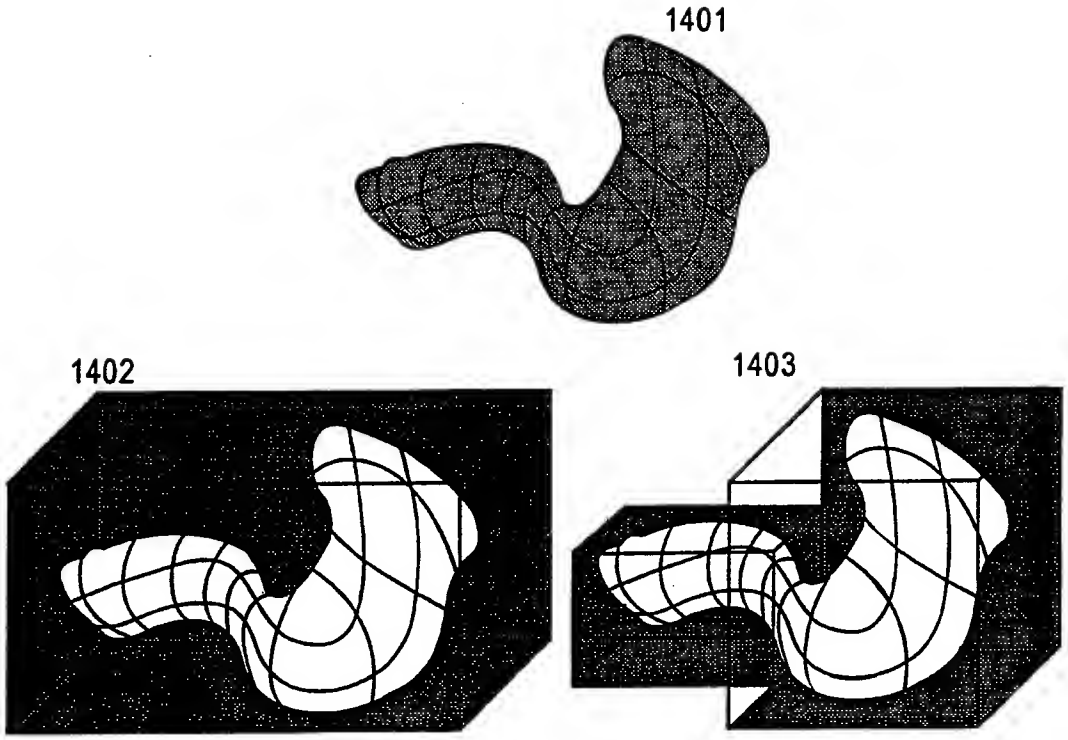


Fig. 14

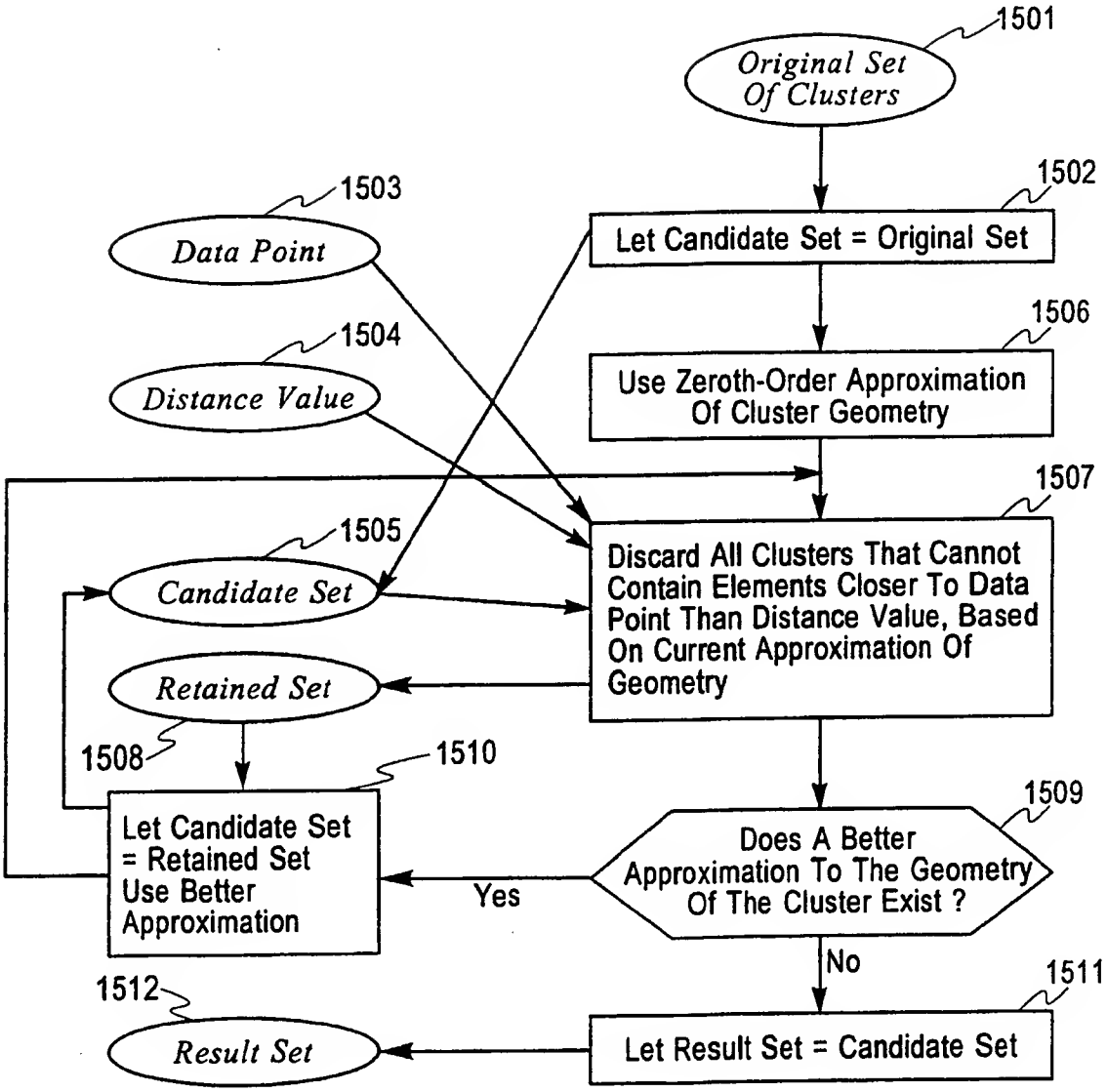

1501
Original Set Of Clusters
1502
Let Candidate Set = Original Set
1503
Data Point
1504
Distance Value
1505
Candidate Set
1506
Use Zeroth-Order Approximation Of Cluster Geometry
1507
Discard All Clusters That Cannot Contain Elements Closer To Data Point Than Distance Value, Based On Current Approximation Of Geometry
Retained Set
1508
1509
Does A Better Approximation To The Geometry Of The Cluster Exist ?
Yes
No
1510
Let Candidate Set = Retained Set Use Better Approximation
1511
Let Result Set = Candidate Set
1512
Result Set

Fig. 15

INTERNATIONAL SEARCH REPORT

in ational Application No
PCT/GB 98/03196

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F17/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	NG R ET AL: "EVALUATING MULTI-DIMENSIONAL INDEXING STRUCTURES FOR IMAGES TRANSFORMED BY PRINCIPAL COMPONENT ANALYSIS" STORAGE AND RETRIEVAL FOR STILL IMAGE AND VIDEO DATABASES 4, SAN JOSE, FEB. 1 - 2, 1996, no. VOL. 2670, 1 February 1996, pages 50-61, XP000642562 SETHI I K;JAIN R C (EDS) see page 52, line 24 - page 53, line 9 ---	1-4,12, 27,28
A	US 5 179 643 A (HOMMA KOICHI ET AL) 12 January 1993 see the whole document ---	1-28
	-/--	

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Data of the actual completion of the international search	Data of mailing of the international search report
20 January 1999	27/01/1999

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Abbing, R

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

In tional Application No
PCT/GB 98/03196

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	US 5 647 058 A (AGRAWAL RAKESH ET AL) 8 July 1997 see abstract see column 3, line 65 - column 4, line 46 see column 5, line 47 - column 6, line 27 see claims	1-3,27, 28
A	"MULTIDIMENSIONAL INDEX STRUCTURE WITH MULTI-LEVEL ENTRY AND SKIP-LEVEL SEARCH FOR PARTIALLY-SPECIFIED QUERIES" IBM TECHNICAL DISCLOSURE BULLETIN, vol. 39, no. 11, November 1996, pages 37-39, XP000679812 see the whole document	1,27,28

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

In tional Application No
PCT/GB 98/03196

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5179643 A	12-01-1993	JP 2213981 A	27-08-1990
		JP 2170269 A	02-07-1990
		JP 2771201 B	02-07-1998
US 5647058 A	08-07-1997	NONE	